

The L^AT_EX 2 _{ε} Sources

Johannes Braams
David Carlisle
Alan Jeffrey
Leslie Lamport
Frank Mittelbach
Chris Rowley
Rainer Schöpf

2022-06-01 Patch level -3

This file is maintained by the L^AT_EX Project team.
Bug reports can be opened (category `latex`) at
<https://latex-project.org/bugs.html>.

Contents

a	ltdirchk.dtx	1
1	L^AT_EX System Dependent Initializations	1
2	Initialization	2
2.1	INITEX	3
2.2	Some bits of 2e	4
3	texsys.cfg	5
3.1	texsys.cfg	6
3.2	UNIX (web2c)	7
3.3	UNIX (other)	7
3.4	MSDOS (emtex)	7
3.5	MSDOS (other)	7
3.6	VMS (DECUS T _E X, PD VMS 3.6)	7
3.7	VMS (???)	8
3.8	MACINTOSH (OzTeX 1.6)	8
3.9	MACINTOSH (other)	8
3.10	FAKE EXAMPLE	8
4	Setting \@currdir	9
5	Setting \input@path	11

6	Filename Parsing	11
7	T_EX Versions	13
8	ltxcheck.tex	14
b	ltplain.dtx	15
1	Plain T_EX	15
c	ltvers.dtx	39
1	Version Identification	39
1.1	Declaring an all-new module	42
d	ltluatex.dtx	44
1	Overview	44
2	Core T_EX functionality	44
3	Plain T_EX interface	45
4	Lua functionality	45
4.1	Allocators in Lua	45
4.2	Lua access to T _E X register numbers	46
4.3	Module utilities	47
4.4	Callback management	47
5	Implementation	48
5.1	Minimum LuaT _E X version	48
5.2	Older L ^A T _E X/Plain T _E X setup	48
5.2.1	Fixes to <i>etex.src/etex.sty</i>	49
5.2.2	luatex specific settings	49
5.3	Attributes	50
5.4	Category code tables	51
5.5	Named Lua functions	52
5.6	Custom whatsis	53
5.7	Lua bytecode registers	53
5.8	Lua chunk registers	54
5.9	Lua loader	54
5.10	Lua module preliminaries	55
5.11	Lua module utilities	56
5.11.1	Module tracking	56
5.11.2	Module messages	57
5.12	Accessing register numbers from Lua	58
5.13	Attribute allocation	59
5.14	Custom whatsit allocation	59

5.15	Bytecode register allocation	60
5.16	Lua chunk name allocation	60
5.17	Lua function allocation	61
5.18	Lua callback management	61
5.18.1	Housekeeping	61
5.18.2	Handlers	64
5.18.3	Public functions for callback management	66
e	ltexpl.dtx	72
1	expl3-dependent code	72
1.1	Loader	72
1.2	Using expl3 code	75
2	Document-level command names for expl3 functions	76
f	ltdefns.dtx	78
1	Definitions	78
1.1	Initex initializations	78
1.2	Saved versions of T _E X primitives	78
1.3	Command definitions	79
1.4	Robust commands and protect	88
1.5	Acting on robust commands	94
1.5.1	Copying robust commands	96
1.5.2	Showing robust commands	97
1.5.3	Commands defined with \DeclareRobustCommand	98
1.5.4	Commands defined with \newcommand (with optional argument) .	100
1.6	Internal defining commands	101
2	Discretionary Hyphenation	105
g	ltcmd.dtx	108
1	Creating document commands	108
1.1	Variables and constants	108
1.2	Declaring commands and environments	111
1.3	Structure of xparse commands	115
1.4	Normalizing the argument specifications	120
1.5	Preparing the signature: general mechanism	128
1.6	Setting up a standard signature	129
1.7	Setting up expandable types	134
1.7.1	Copying a command and its internal structure	137
1.7.2	Showing the definition of a command	142
1.8	Grabbing arguments	146
1.9	Grabbing arguments expandably	158
1.10	Argument processors	163
1.11	Access to the argument specification	165

1.12	Utilities	167
1.13	Messages	171
1.14	User functions	176
h	lthooks.dtx	181
1	Introduction	181
2	Package writer interface	181
2.1	$\text{\LaTeX} 2\epsilon$ interfaces	181
2.1.1	Declaring hooks	181
2.1.2	Special declarations for generic hooks	182
2.1.3	Using hooks in code	182
2.1.4	Updating code for hooks	183
2.1.5	Hook names and default labels	185
2.1.6	The <code>top-level</code> label	187
2.1.7	Defining relations between hook code	188
2.1.8	Querying hooks	189
2.1.9	Displaying hook code	190
2.1.10	Debugging hook code	191
2.2	L3 programming layer (<code>expl3</code>) interfaces	191
2.3	On the order of hook code execution	193
2.4	The use of “reversed” hooks	195
2.5	Difference between “normal” and “one-time” hooks	196
2.6	Generic hooks provided by packages	197
2.7	Private \LaTeX kernel hooks	197
2.8	Legacy $\text{\LaTeX} 2\epsilon$ interfaces	198
3	$\text{\LaTeX} 2\epsilon$ commands and environments augmented by hooks	199
3.1	Generic hooks	199
3.1.1	Generic hooks for all environments	199
3.1.2	Generic hooks for commands	200
3.1.3	Generic hooks provided by file loading operations	201
3.2	Hooks provided by <code>\begin{document}</code>	201
3.3	Hooks provided by <code>\end{document}</code>	201
3.4	Hooks provided by <code>\shipout</code> operations	203
3.5	Hooks provided for paragraphs	203
3.6	Hooks provided in NFSS commands	203

4	The Implementation	204
4.1	Debugging	204
4.2	Borrowing from internals of other kernel modules	204
4.3	Declarations	205
4.4	Providing new hooks	207
4.4.1	The data structures of a hook	207
4.4.2	On the existence of hooks	208
4.4.3	Setting hooks up	209
4.4.4	Disabling and providing hooks	211
4.5	Parsing a label	213
4.6	Adding or removing hook code	217
4.7	Setting rules for hooks code	226
4.8	Specifying code for next invocation	240
4.9	Using the hook	241
4.10	Querying a hook	244
4.11	Messages	246
4.12	L ^A T _E X 2 _E package interface commands	248
4.13	Deprecated that needs cleanup at some point	251
4.14	Internal commands needed elsewhere	252
i	ltcmdhooks.dtx	254
1	Introduction	254
2	Restrictions and Operational details	255
2.1	Patching	255
2.1.1	Timing	255
2.2	Commands that look ahead	255
3	Package Author Interface	256
4	The Implementation	257
4.1	Execution plan	257
4.2	Variables	258
4.3	Variants	259
4.4	Patching or delaying	259
4.5	Patching commands	260
4.5.1	Patching by expansion and redefinition	261
4.5.2	Patching by retokenization	268
4.6	Messages	272
j	ltalloc.dtx	273
1	Counters	273
k	ltcntrl.dtx	275
1	Program control structure	275

l	lterror.dtx	279
1	Error handling and tracing	279
1.1	General commands	279
1.2	Specific errors	285
1.3	Tracing	289
m	ltpar.dtx	290
1	Paragraphs	290
1.1	Implementation	290
n	ltpara.dtx	292
1	Introduction	292
1.1	The default processing done by the engine	292
2	The new mechanism implemented for L^AT_EX	294
2.1	The provided hooks	295
2.2	Altered and newly provided commands	296
2.3	Examples	297
2.3.1	Testing the mechanism	297
2.3.2	Mark the first paragraph of each <code>itemize</code>	299
2.4	Some technical notes	299
2.4.1	Glue items between paragraphs (found with <code>fancypar</code>)	299
3	The Implementation	300
3.1	Providing hooks for paragraphs	300
3.2	The error messages	306
o	ltmeta.dtx	308
1	Introduction	308
1.1	<code>\DocumentMetadata</code>	308
2	The Implementation	308
p	ltspace.dtx	310
1	Spacing	310
1.1	User Commands	310
1.2	Chris' comments	310
1.3	Some immediate actions	312
1.4	The code	313
1.5	Vertical spacing	320
1.6	Horizontal space (and breaks)	325

q ltlogos.dtx	329
1 Logos	329
r ltfles.dtx	330
1 File Handling	330
1.1 Safe Input Macros	342
1.2 Listing files	349
s ltoutenc.dtx	351
1 Font encodings	351
1.1 Removing encoding-specific commands	353
1.2 The order of declarations	354
1.3 Docstrip modules	354
1.4 Definitions for the kernel	355
1.4.1 Declaration commands	355
1.4.2 Hyphenation	363
1.4.3 Miscellania	363
1.4.4 Default encodings	363
1.4.5 Math material	366
1.5 Definitions for the OT1 encoding	367
1.6 Definitions for the T1 encoding	369
1.7 Definitions for the OMS encoding	375
1.8 Definitions for the OML encoding	376
1.9 Definitions for the OT4 encoding	376
1.10 Definitions for the TS1 encoding	378
1.11 Definitions for the TU encoding	383
2 Package files	394
2.1 The fontenc package	394
t ltcounds.dtx	397
1 Counters and Lengths	397
1.1 Environment Counter Macros	397
u ltlength.dtx	405
1 Lengths	405
v ltfssbas.dtx	407
1 Preliminary macros	407

2	Macros for setting up the tables	408
3	Selecting a new font	415
3.1	Macros for the user	415
3.2	Macros for loading fonts	419
4	Assigning math fonts to <i>versions</i>	427
w	ltfssaxes.dtx	434
1	Changing the font series	434
1.1	The series lookup table	434
1.2	Mapping rules for series changes	435
1.3	Changing to a new series	443
2	Changing the shape	448
2.1	Mapping rules for shape combinations	449
2.2	Changing to a new shape	451
3	Make sure we win . . .	453
x	ltfsstrc.dtx	455
1	Introduction	455
2	A driver for this document	455
3	The Implementation	456
4	Handling Options	456
5	Macros common to <i>fam.tex</i> and <i>tracefnt.sty</i>	458
5.1	General font loading	458
5.2	Math fonts setup	464
5.2.1	Outline of algorithm for math font sizes	464
5.2.2	Code for math font size setting	465
5.2.3	Other code for math	466
6	Scaled font extraction	468
6.1	Sizefunctions	476
y	ltfsscmp.dtx	480
z	ltfssdcl.dtx	485
1	Interface Commands	485

A	ltfssini.dtx	515
1	NFSS Initialization	515
1.1	Providing math <i>versions</i>	515
2	Custom series settings for main document families	516
3	Supporting nested emphasis	533
3.1	Legacy	537
3.2	Miscellaneous	537
B	fontdef.dtx	543
1	Introduction	543
2	Customization	543
3	The <code>docstrip</code> modules	544
4	A driver for this document	544
5	The <code>fonttext.ltx</code> file	544
5.1	Encodings	545
5.2	Defaults	547
6	The <code>fontmath.ltx</code> file	549
6.1	The font encodings used	549
6.1.1	Symbolfont and Alphabet declarations	549
6.2	Math font sizes	550
6.3	The math symbol assignments	551
6.3.1	The letters	551
6.3.2	The digits	552
6.3.3	Punctuation, brace, etc. keys	552
6.3.4	Delimtercodes for characters	552
6.4	Symbols accessed via control sequences	553
6.4.1	Greek letters	553
6.4.2	Ordinary symbols	554
6.4.3	Large Operators	554
6.4.4	Binary symbols	555
6.4.5	Relations	556
6.4.6	Arrows	557
6.4.7	Punctuation symbols	558
6.4.8	Math accents	559
6.4.9	Radicals	559
6.4.10	Over and under something, etc	559
6.4.11	Delimiters	560
6.5	Math versions of text commands	561
6.6	Other special functions and parameters	561
6.6.1	Biggggg	561
6.6.2	The log-like functions	562

6.6.3	Parameters	562
7	Default cfg files	562
C	preload.dtx	564
1	Overview	564
2	Customization	564
3	Module switches for the DOCSTRIP program	564
4	A driver for this document	565
5	The code	565
D	ltfntcmd.dtx	567
1	Introduction	567
2	The implementation	569
3	Initialization	575
E	lttextcomp.dtx	576
1	Sub-encodings	580
1.1	Unavailable in sub-encoding 1 and higher (drop symbols not working in Latin Modern)	581
1.2	Unavailable in sub-encoding 2 (majority of new OTF fonts via autoinst) and higher	581
1.3	Unavailable in sub-encoding 3 and higher	585
1.4	Unavailable in sub-encoding 4 and higher	585
1.5	Unavailable in sub-encoding 5 (most older PS fonts) and higher	585
1.6	Unavailable in sub-encoding 6 and higher	586
1.7	Unavailable in sub-encoding 7 and higher	586
1.8	Unavailable in sub-encoding 8 and higher	586
1.9	Unavailable in Sub-encoding 9 (most missing)	586
2	Unicode engine specials	586
3	Font family sub-encodings setup	587
4	Legacy symbol support for lists and footnote symbols	591
5	The textcomp package	596
5.1	The old textcomp package code	597
5.1.1	Supporting oldstyle digits	605
5.1.2	Subset encoding defaults	606

F	ltpageno.dtx	608
1	Page Numbering	608
G	ltxref.dtx	609
1	Cross Referencing	609
1.1	Cross Referencing	609
H	ltmiscen.dtx	614
1	Miscellaneous Environments	614
1.1	Environments	614
1.2	Center, Flushright, Flushleft	626
1.3	Verbatim	629
I	ltmath.dtx	635
1	Math setup	635
1.1	Math commands based on plain TeX	635
1.1.1	The log-like functions	635
1.1.2	Biggggg	636
1.1.3	The UNSORTED Rest	636
1.2	Math Environments	642
1.3	External options to the standard document classes	647
1.3.1	Left equation numbering	647
1.3.2	Flush left equations	647
J	ltlists.dtx	650
1	List, and related environments	650
1.1	List and Trivlist	651
1.2	Vertical Spacing (skips)	652
1.3	Penalties	652
1.4	Horizontal Spacing (dimens)	652
1.5	Default Values	652
1.6	Itemize and Enumerate	663
K	ltboxes.dtx	666
1	L ATEX Box commands	666
1.1	Some low-level constructs	681
L	lttab.dtx	682

1	Tabbing, Tabular and Array Environments	682
1.1	tabbing	682
1.2	array and tabular environments	691
M	ltpictur.dtx	707
1	Picture Mode	707
1.1	Curves	734
N	ltthm.dtx	739
1	Theorem Environments	739
O	ltsect.dtx	743
1	Sectioning Commands	743
1.1	The Title	743
1.2	Sectioning	744
1.2.1	Initializations	751
1.3	Table of Contents etc.	751
1.3.1	Convention	751
1.3.2	Commands	751
P	ltfloat.dtx	756
1	Floats	756
1.1	Floating Environments	756
1.2	Footnotes	770
Q	ltidxglo.dtx	779
1	Index and Glossary Generation	779
R	ltbibl.dtx	782
1	Bibliography Generation	782
1.1	Default definitions	786
S	ltpage.dtx	787
1	Page styles and related commands	787
1.1	Page Style Commands	787
1.2	How a page style makes running heads and feet	787
1.3	marking conventions	787

T	ltclass.dtx	791
1	Introduction	791
2	User interface	791
2.1	Option processing	792
3	Class and Package interface	793
3.1	Class name and version	793
3.2	Package name and version	793
3.3	Requiring other packages	793
3.4	Declaring new options	794
3.5	Safe Input Macros	795
4	Implementation	795
4.1	Hooks	822
4.2	Providing shipment	824
5	Package/class rollback mechanism	832
6	After Preamble	840
U	ltkeys.dtx	841
1	Creating and using keyval options	841
1.1	Implementation of ltkeys	842
1.2	Key properties	842
1.3	Main mechanism	843
1.4	The document interfaces	845
1.5	Option usage scope	846
1.6	General key setting	847
V	ltfilehook.dtx	848
1	Introduction	848
1.1	Provided hooks	848
1.2	General hooks for file reading	848
1.3	Hooks for package and class files	849
1.4	Hooks for \include files	850
1.5	High-level interfaces for L ^A T _E X	851
1.6	Internal interfaces for L ^A T _E X	852
1.7	A sample package for structuring the log output	852

2	The Implementation	853
2.1	Document and package-level commands	853
2.2	expl3 helpers	854
2.3	Declaring the file-related hooks	857
2.4	Patching L ^A T _E X's \InputIfFileExists command	857
2.5	Declaring a file substitution	859
2.6	Selecting a file (\set@curr@file)	861
2.7	Replacing a file and detecting loops	864
2.7.1	The Tortoise and Hare algorithm	865
2.8	Preventing a package from loading	867
2.9	High-level interfaces for L ^A T _E X	867
2.10	Internal commands needed elsewhere	868
3	A sample package for structuring the log output	869
4	Package emulations	870
4.1	Package atveryend emulation	870
W	ltshipout.dtx	871
1	Introduction	871
1.1	Overloading the \shipout primitive	871
1.2	Provided hooks	872
1.3	Legacy L ^A T _E X commands	874
1.4	Special commands for use inside the hooks	874
1.5	Provided LuaT _E X callbacks	875
1.6	Information counters	875
1.7	Debugging shipout code	876
2	Emulating commands from other packages	876
2.1	Emulating atbegshi	876
2.2	Emulating everyshi	877
2.3	Emulating atenddvi	877
2.4	Emulating everypage	878
3	The Implementation	878
3.1	Debugging	878
3.2	Handling the end of job hook	890
4	Legacy L^AT_EX 2_ε interfaces	893
5	Internal commands needed elsewhere	894
6	Package emulation for compatibility	895
6.1	Package atenddvi emulation	895
6.2	Package atbegshi emulation	896
6.3	Package everyshi emulation	897
X	ltoutput.dtx	898

1	Output Routine	898
1.1	Floats	898
1.1.1	Kludgeins	953
1.1.2	Float control	956
1.1.3	Float placement parameters	968
Y	lthyphen.dtx	972
Z	ltfinal.dtx	974
1	Final settings	974
1.1	Debugging	974
1.2	Typesetting parameters	974
1.3	Lccodes for hyphenation	977
1.4	Hyphenation	980
1.5	Font loading	981
1.6	Input encoding	982
1.7	Lccodes and uccodes	986
1.8	Applying Patch files	988
1.9	Freeing Memory	989
1.10	Initialise file list	990
1.11	Preparation for supporting PDF in backends	990
1.12	Do some temporary work for pre-release	990
1.13	Some last minute initializations	991
1.14	Dumping the format	991
Change History		992
Index		1061

File a

ltdirchk.dtx

1 L^AT_EX System Dependent Initializations

This file implements the semi-automatic determination of various system dependent parts of the initialization. The actual definitions may be placed in a file `texsys.cfg`. Thus for operating systems for which the tests here do not result in acceptable settings, a ‘hand written’ `texsys.cfg` may be produced.

The macros that must be defined are:

`\@currdir` `\@currdir{filename}{space}` should expand to a form of the filename that uniquely refers to the ‘current directory’ if this is possible. (The expansion should also end with a space.) on UNIX, this is `\def\@currdir{./}`. For more exotic operating systems you may want to make `\@currdir` a macro with arguments delimited by . and/or `{space}`. If the operating system has no concept of directory structure, this macro should be defined to be empty.

`\input@path` If the primitive `\openin` searches the same directories as the primitive `\input`, then it is possible to tell (using `\ifeof`) whether a file exists before trying to input it. For systems like this, `\input@path` should be left undefined.

If `\openin` does not ‘follow’ `\input` then `\input@path` must be defined to be a list of directories to search for input files. The format for each directory is as for `\@currdir`, normally just a prefix is required, but it may be a macro with space-delimited argument. That is, if `{dir}` is an entry in the input path, T_EX will try to load the expansion of `{dir}{filename}{space}`

So either `{dir}` should be defined as a macro with argument delimited by space, or it should just expand to a directory name, including the final directory separator, so that it may be concatenated with the `{filename}`. This means that for UNIX-like syntax, each `{dir}` should end with a slash, /.

`\input@path` should expand to a list of such directories, each in a {} group.

`\filename@parse` After a call of the form: `\filename@parse{filename}`, the three macros `\filename@area`, `\filename@base` and `\filename@ext` should be defined to be the ‘area’ (or directory), basename and extension respectively. If there was no extension specified in `{filename}`, `\filename@ext` should be `\let` to `\relax` (so this case may be tested with `\@ifundefined{\filename@ext}` and, perhaps a default extension substituted).

Normally one would not need to define this macro in `texsys.cfg` as the automatic tests can supply parsers that work with UNIX and VMS and Macintosh syntax, as well as a basic parser that will cover many other cases. However some operating systems may need a ‘hand produced’ parser in which case it should be defined in this file.

The UNIX parser also works for most MSDOS T_EX versions. Currently if the UNIX, VMS or Macintosh parser is not used, `\filename@parse` is defined to always return an empty area, and to split the argument into basename and extension at the first ‘.’ that occurs in the name. Parsers for other formats may be defined in `texsys.cfg`, in which case they will be used in preference to the default definitions.

`\@TeXversion` `\@TeXversion` is now set automatically by the initialization tests in this file. You should not need to set it in `texsys.cfg`, however the following documentation is left for information. L^AT_EX does not set this variable exactly, the automatic tests set it to:
2 for any version, v , $v < 3.0$
3 for any version, v , $3.0 \leq v \leq 3.14$

<undefined> otherwise.

However these values are accurate enough for L^AT_EX to take appropriate action for these old T_EXs.

If your T_EX is older than version 3.141, then you should define `\@TeXversion` (using `\def`) to be the version number. If you do not do this¹, L^AT_EX will not work around a bug in old T_EX versions, and so error messages will appear in a very strange format, with `^J` appearing instead of line breaks:

```
LaTeX Error: \rubbish undefined.^J^JSee the LaTeX manual or LaTeX=
Companion
for explanation.^JType H <return> for immediate help.
...
.3 \renewcommand{\rubbish}
{}
```

However if you put `\def\@TeXversion{3.14}` in `texsys.cfg` the following format will be used:

```
LaTeX Error: \rubbish undefined.

See the LaTeX manual or LaTeX Companion for explanation.
Type H <return> for immediate help.
.
...
.3 \renewcommand{\rubbish}
{}
```

Note that this has an extra line `! .` which does not appear in error messages that use the default settings with a current version of T_EX, but this should not cause any confusion we hope.

2 Initialization

As this file is read at a very early stage, some definitions that are normally considered to be part of the format must be made here.

Most such definitions are repeated later in the “right” place, usually (but not always) with different implementations. To be able to spot this more easily if you look into the file `latex.ltx` (which is stripped of comments) we add some comment lines to that effect that survive the stripping process by `docstrip`.

```
1 <*dircheck>
2 %% ---- START temporary definitions for bootstrapping; later overwritten ---
3 </dircheck>
```

¹Actually if your T_EX is really old, version 2, L^AT_EX can detect this, and sets `\@TeXversion` to 2 if it is not set in the `cfg` file.

2.1 INITEX

```

4  <*dircheck>
5  <*initex>
6  <initex> \ifnum\catcode`\\=1
7  <initex>   \errmessage
8  <initex>   {LaTeX must be made using an initex with no format preloaded}
9  <initex> \fi
10 \catcode`\\=1
11 \catcode`\\=2

```

If LuaTeX is in use the extensions and other new primitives have to be activated: this is done as early as possible. Older versions of LuaTeX do not hide the primitives: a version check is not needed as the version itself will be missing in the case where action is needed!

```

12 \ifx\directlua\undefined
13 \else
14   \ifx\luatexversion\undefined

```

Enable e-TeX/pdfTeX/Umath primitives with their natural names

```

15   \directlua{tex.enableprimitives("",%
16             tex.extraprimitives('etex', 'pdftex', 'umath'))}

```

In current formats enable primitives with unprefixed names. the `\textralaterelease` guards allow the primitives to be defined with a `\luatex` prefix if older formats are specified.

```

17 </initex>
18 </dircheck>
19 <*initex, \textralaterelease>
20 <\textralaterelease> \ifx\directlua\undefined\else
21 <\textralaterelease> \IncludeInRelease{2015/10/01}{\luatexluafunction}
22 <\textralaterelease>                               {LuaTeX (prefixed names)}%
23   \directlua{tex.enableprimitives("",%
24             tex.extraprimitives("omega", "aleph", "luatex"))}
25 <\textralaterelease> \EndIncludeInRelease
26 <\textralaterelease> \IncludeInRelease{0000/00/00}{\luatexluafunction}
27 <\textralaterelease>                               {LuaTeX (prefixed names)}%
28 <\textralaterelease> \directluaf
29 <\textralaterelease>   tex.enableprimitives(
30 <\textralaterelease>     "luatex",
31 <\textralaterelease>     tex.extraprimitives("core", "omega", "aleph", "luatex")
32 <\textralaterelease>   )
33 <\textralaterelease>   local i
34 <\textralaterelease>   local t = { }
35 <\textralaterelease>   for _,i in pairs(tex.extraprimitives("luatex")) do
36 <\textralaterelease>     if not string.match(i,"^U") then
37 <\textralaterelease>       if not string.match(i, "^luatex") then
38 <\textralaterelease>         table.insert(t,i)
39 <\textralaterelease>       end
40 <\textralaterelease>     else
41 <\textralaterelease>       if string.match(i,"^Uchar$") then
42 <\textralaterelease>         table.insert(t,i)
43 <\textralaterelease>       end
44 <\textralaterelease>     end
45 <\textralaterelease>   end
46 <\textralaterelease>   for _,i in pairs(t) do
47 <\textralaterelease>     tex.print(
48 <\textralaterelease>       "\noexpand\\let\\noexpand\\" .. i

```

```

49 <{latexrelease}          .. "\noexpand\undefined"
50 <{latexrelease}      )
51 <{latexrelease}  end
52 <{latexrelease} }
53 <{latexrelease}\EndIncludeInRelease
54 <{latexrelease}\fi
55 </initex, latexrelease>
56 <{*dircheck}
57 <{*initex}
58   \fi
59 \fi

A test can now be made for eTeX.
60 <{initex}\ifx\TeXversion\undefined
61 <{initex}  \errmessage
62 <{initex}    {LaTeX requires e-Tex}
63 <{initex}  \expandafter\endinput
64 <{initex}\fi

That distraction over, back to the basics of a format.
65 \catcode`#=6
66 \catcode`^=7
67 \chardef\active=13
68 \catcode`\@=11
69 \countdef\count@=255
70 \let\bgroup={ \let\egroup=}
71 \ifx\@@input\@undefined\let\@@input\input\fi
72 \ifx\@@end\@undefined\let\@@end\end\fi
73 \chardef\@inputcheck0
74 \chardef\sixt@n=16
75 \newlinechar`\^\J
76 \def\typeout{\immediate\write17}
77 \def\dospecials{\do\ \do\\\do{\{}{\do{\}}\do\$\\do\&%
78   \do#\do\^\do_\\do%\do\~}
79 \def\@makeother#1{\catcode`#1=12\relax}
80 \def\space{ }
81 \def\@tempswafalse{\let\if@tempswa\iffalse}
82 \def\@tempswatrue{\let\if@tempswa\iftrue}
83 \let\if@tempswa\iffalse
84 \def\loop#1\repeat{\def\iterate{\#1\relax\expandafter\iterate\fi}%
85   \iterate \let\iterate\relax}
86 \let\repeat\fi
87 </initex>

```

2.2 Some bits of 2e

```

88 <{*2ekernel}
89 \def\two@digits#1{\ifnum#1<10 0\fi\number#1}
90 \long\def\@firstoftwo#1#2{#1}
91 \long\def\@secondoftwo#1#2{#2}

```

This is a special version of \ProvidesFile for initex use.

```

92 \def\ProvidesFile#1{%
93   \begingroup
94     \catcode`\ 10 %
95     \ifnum \endlinechar<256 %

```

```

96      \ifnum \endlinechar>\m@ne
97          \catcode\endlinechar 10 %
98      \fi
99      \fi
100     \makeother\%
101     \ifnextchar[{\@providesfile{#1}}{\@providesfile{#1}[]}
102     \def\@providesfile#1[#2]{%
103         \wlog{File: #1 #2}%
104         \@addtofilelist{ #2}%
105         \endgroup}
106     \long\def\@addtofilelist#1{%
107     \def\@empty{}%
108     \catcode`\%=12%
109     \def\@percentchar{\%}%
110     \catcode`\%=14%
111     \let\@currdir\@undefined%
112     \let\input@path\@undefined%
113     \let\filename@parse\@undefined%
\nstrip@prefix
114     \def\strip@prefix#1{}%
115     </2ekernel>

```

(End definition for `\strip@prefix`.)

3 texsys.cfg

As mentioned above, any site specific definitions required to describe the filename handling must be entered into a file `texsys.cfg`. If `texsys.cfg` can not be located by `\openin`, we write a default version out. The default version only contains comments, so we do not actually input the file in that case. The automatic tests later will, hopefully, correctly define the required macros.

The tricky code below checks to see if `texsys.cfg` exists. If it does not, all the text in this file between START and END is copied verbatim to a new file `texsys.cfg`. If `texsys.cfg` is found, then it is simply input. This is only done when this file is being used unstripped.

```

116  {*docstrip}
117  \openin15=texsys.cfg
118  \ifeof15
119  \typeout{** Writing a default texsys.cfg}
120  \immediate\openout15=texsys.cfg
121  \begingroup
122  \catcode`^=M\active%
123  \let^=M\par%
124  \def\reserved@a#1^=M{%
125  \def\reserved@b{#1}%
126  \ifx\reserved@b\reserved@c\endgroup\else%
127  \immediate\write15{#1}%
128  \expandafter\reserved@a\fi}%
129  \def\reserved@d#1START^=M{\let\do\@makeother\dospecials\reserved@a}%
130  \catcode`\%=12%
131  \def\reserved@c{\%END}%
132  \reserved@d

```

START

3.1 texsys.cfg

This file contains the site specific definitions of the four macros `\@currdir`, `\input@path`, `\filename@parse` and `\@TeXversion`.

As distributed it only contains comments, however this ‘empty’ file will work on many systems because of the automatic tests built into `ltdirchk.dtx`. You are allowed to edit this file to add definitions of these macros appropriate to your system.

The macros that must be defined are:

`\@currdir` `\@currdir<filename><space>` should expand to a form of the filename that uniquely refers to the ‘current directory’ if this is possible. (The expansion should also end with a space.) on UNIX, this is `\def\@currdir{./}`. For more exotic operating systems you may want to make `\@currdir` a macro with arguments delimited by . and/or `<space>`. If the operating system has no concept of directory structure, this macro should be defined to be empty.

`\input@path` If the primitive `\openin` searches the same directories as the primitive `\input`, then it is possible to tell (using `\ifeof`) whether a file exists before trying to input it. For systems like this, `\input@path` should be left undefined.

If `\openin` does not ‘follow’ `\input` then `\input@path` must be defined to be a list of directories to search for input files. The format for each directory is as for `\@currdir`, normally just a prefix is required, but it may be a macro with space-delimited argument. That is, if `<dir>` is an entry in the input path, TeX will try to load the expansion of `<dir><filename><space>`

So either `<dir>` should be defined as a macro with argument delimited by space, or it should just expand to a directory name, including the final directory separator, so that it may be concatenated with the `<filename>`. This means that for UNIX-like syntax, each `<dir>` should end with a slash, /. One exception to this rule is that the input path should always contain the empty directory {} as this will allow ‘full pathnames’ to be used, and the ‘current directory’ to be searched.

`\input@path` should expand to a list of such directories, each in a {} group.

`\filename@parse` After a call of the form: `\filename@parse{<filename>}`, the three macros `\filename@area`, `\filename@base`, `\filename@ext` should be defined to be the ‘area’ (or directory), basename and extension respectively. If there was no extension specified in `<filename>`, `\filename@ext` should be `\let` to `\relax` (so this case may be tested with `\@ifundefined{\filename@ext}` and, perhaps a default extension substituted).

Normally one would not need to define this macro in `texsys.cfg` as the automatic tests can supply parsers that work with UNIX and VMS syntax, as well as a basic parser that will cover many other cases. However some operating systems may need a ‘hand produced’ parser in which case it should be defined in this file.

The UNIX parser also works for most MSDOS TeX versions. Currently if the UNIX or VMS parser is not used, `\filename@parse` is defined to always return an empty area, and to split the argument into basename and extension at the first ‘.’ that occurs in the name. Parsers for other formats may be defined in `texsys.cfg`, in which case they will be used in preference to the default definitions.

`\@TeXversion` You should not need to set this macro in `texsys.cfg`. LATEX tests to set this automatically. See the comments in the opening section of `ltdirchk.dtx`.

The following sections give examples of definitions which might work on various systems. These are currently mainly untested as I only have access to a few systems, all

of which do not need this file as the automatic tests work. All the code is commented out.

3.2 UNIX (web2c)

This implementation does make `\openin` and `\input` look in the same places. Acceptable settings are made by `ltdirchk.dtx`, and so this file may be empty. The definitions below are therefore just for information.

```
133 % \def\@currdir{./}
134 % \let\input@path\@undefined
```

3.3 UNIX (other)

Apparently some commercial UNIX implementations have different paths for `\openin` and `\input`. For these one could use definitions like the following (with whatever directories are used at your site): note that the directory names should end with `/`.

```
135 % \def\@currdir{./}
136 % \def\input@path{%
137 %   {/usr/local/lib/tex/inputs/distrib/}%
138 %   {/usr/local/lib/tex/inputs/contrib/}%
139 %   {/usr/local/lib/tex/inputs/local/}%
140 % }
```

3.4 MSDOS (emtex)

This implementation does make `\openin` and `\input` look in the same places. Acceptable settings are made by `ltdirchk.dtx`, and so this file may be empty. The definitions below are therefore just for information.

```
141 % \def\@currdir{./}
142 % \let\input@path\@undefined
```

3.5 MSDOS (other)

Some PC implementations have different paths for `\openin` and `\input`. For these one could use definitions like the following (with whatever directories are used at your site): note that the directory names should end with `/`. This assumes the implementation uses UNIX style `/` as the directory separator.

```
143 % \def\@currdir{./}
144 % \def\input@path{%
145 %   {c:/tex/inputs/distrib/}%
146 %   {c:/tex/inputs/contrib/}%
147 %   {c:/tex/inputs/local/}%
148 % }
```

3.6 VMS (DECUS T_EX, PD VMS 3.6)

This implementation does make `\openin` and `\input` look in the same places. Acceptable settings are made by `ltdirchk.dtx`, and so this file may be empty. The definitions below are therefore just for information.

```
149 % \def\@currdir{[]}
150 % \let\input@path\@undefined
```

3.7 VMS (???)

Some VMS implementations have different paths for `\openin` and `\input`. For these one could use definitions like the following:

```
151 % \def\@currdir{[]}
152 % \def\input@path{%
153 %   {tex_inputs:}%
154 %   {SOMEDISK:[SOME.TEX.DIRECTORY]}%
155 % }
```

3.8 MACINTOSH (OzTeX 1.6)

This implementation does make `\openin` and `\input` look in the same places. Acceptable settings are made by `ltdirchk.dtx`, and so this file may be empty. The definitions below are therefore just for information.

```
156 % \def\@currdir{::}
157 % \let\input@path\@undefined
```

3.9 MACINTOSH (other)

Some Macintosh implementations have different paths for `\openin` and `\input`. For these one could use definitions like the following (with whatever folders are used on your machine): note that the directory names should end with `:`, and they should contain *no* spaces.

```
158 % \def\@currdir{::}
159 % \def\input@path{%
160 %   {Hard-Disk:Applications:TeX:TeX-inputs:}%
161 %   {Hard-Disk:Applications:TeX:My-inputs:}%
162 % }
```

3.10 FAKE EXAMPLE

This example is for an operating system that has filenames of the form `<area>name`. For maximum compatibility with macro sets, you want `name.ext` to be mapped to `<ext>name`, and `<area>name.ext` to be mapped to `<area.ext>name`. `\input` does this mapping automatically, but `\openin` does not, and does not look in the same places as `\input`. `<>name` is the desired ‘current directory’ syntax.

the following code would possibly work:

```
163 % \def\@dir#1#2 {%
164 %   \@dcr{#1}#2..\@nil%
165 % \def\@dcr#1#2.#3.#4\@nil{%
166 %   <\ifx\@dir#1\@dir\else#1\ifx\@dir#3\@dir\else.\fi\fi#3>#2 %
167 %
168 % \def\@currdir{\@dir{}}
169 % \def\input@path{%
170 %   {\@dir{area.one}}%
171 %   {\@dir{area.two}}%
172 % }
```

END

```
173 \immediate\closeout15
```

If `texsys.cfg` did exist, then input it.

```
174 \else
175 \typeout{** Using the existing texsys.cfg}
176 \closein15
177 \input texsys.cfg
178 \fi
179 \{/docstrip\}
```

If the stripped version of this file is being used (in `latex2e.ltx`) then `texsys.cfg` should be there, so just input it.

```
180 \dircheck\input texsys.cfg
```

4 Setting `\@currdir`

`\@currdir` This is a local definition of `\IfFileExists`. It tries to relocate `texsys.aux`. If it succeeds, then the `\@currdir` syntax has been determined. If all the tests fail then `\@currdir` will be set to `\@empty`, and `ltxcheck` will warn of this when it checks the format.

```
181 \begingroup
182 \count@\time
183 \divide\count@ 60
184 \count2=-\count@
185 \multiply\count2 60
186 \advance\count2 \time
```

The current date and time stamp.

```
187 \edef\today{%
188   \the\year/\two@digits{\the\month}/\two@digits{\the\day}:
189   \two@digits{\the\count@}:\two@digits{\the\count2}}
```

Create a file `texsys.aux` (hopefully in the current directory), then try to locate it again.

```
190 \immediate\openout15=texsys.aux
191 \immediate\write15{\today^J}
192 \immediate\closeout15 %
```

#1 is the file to try, #2 is what to do on success, #3 on failure. Note that this definition is overwritten later on again!

```
193 \def\IfFileExists#1#2#3{%
194   \openin\@inputcheck#1 %
195   \ifeof\@inputcheck
196     #3\relax
197   \else
198     \read\@inputcheck to \reserved@a
199     \ifx\reserved@a\today
200       \typeout{#1 found}#2\relax
201     \else
202       \typeout{BAD: old file \reserved@a (should be \today)}%
203       #3\relax
204     \fi
205   \fi
206   \closein\@inputcheck}
```

```

207 \endlinechar=-1
If \@currdir has not been pre-defined in texsys.cfg then test for UNIX, VMS and
Oz-TEX-Mac. syntax.
208 \ifx\@currdir\@undefined
209   \IfFileExists{./texsys.aux}{\gdef\@currdir{./}}%
210   {\IfFileExists{[]texsys.aux}{\gdef\@currdir{[]}}%
211   {\IfFileExists{texsys.aux}{\gdef\@currdir{}}{}}

```

If it is still undefined at this point, all the above tests failed. Earlier versions interactively prompted for a definition at this point, but it seems impossible to reliably obtain information from users at this point in the installation. This version of the file produces a format with no user-interaction. Later if the format is not suitable for the system, **texsys.cfg** may be edited and the format re-made.

```

212 \ifx\@currdir\@undefined
213   \global\let\@currdir\@empty
214   \typeout{\string^J^J%
215     !! No syntax for the current directory could be found\string^J%
216   }%
217 \fi

```

Otherwise \@currdir was defined in **texsys.cfg**. In this case check that the syntax specified works on this system. (In case a complete **LATEX** system has been copied from one system to another.) If the test fails, give up. The installer should remove or correct the offending **texsys.cfg** and try again.

```

218 \else
219   \IfFileExists{@currdir texsys.aux}{}{%
220     \edef\reserved@a{\errhelp{%
221       texsys.cfg specifies the current directory syntax to be\string^J%
222       \meaning\@currdir\string^J%
223       but this does not work on this system.\string^J%
224       Remove texsys.cfg and restart.}}\reserved@a
225     \errmessage{Bad texsys.cfg file: \noexpand\@currdir}\@@end}

```

The version of \@currdir in **texsys.cfg** looks OK.

```

226 \fi
227 \immediate\closeout15 %
228 \endgroup
229 \typeout{\string^J^J%
230   \noexpand\@currdir set to:
231   \expandafter\strip@prefix\meaning\@currdir.\string^J%
232 }

```

(End definition for \@currdir, \IfFileExists, and \today.)

Stop here if the file is being used unstripped.

```

233 <*docstrip>
234 \relax\endinput
235 </docstrip>

```

5 Setting \input@path

Earlier versions of this file attempted to automatically test whether `\input@path` was required, and interactively prompt for a path if necessary. This was not found to be very reliable. The first-time installer of L^AT_EX 2_E can not be expected to have enough information to supply the correct information to the prompts. Now the interaction is omitted. After the format is made the installer can attempt to run the test document `ltxcheck.tex` through L^AT_EX 2_E. This will check, among other things, whether `texsys.cfg` will need to be edited and the format remade.

`\input@path` Now set up the `\input@path`.

`\input@path` should either be undefined, or a list of directories as described in the introduction.

```
236   \typeout{^^J%
237     Assuming \noexpand\openin and \noexpand\input^^J%
238     \ifx\input@path\@undefined
```

`\input@path` has not been pre-defined.

```
239     have the same search path.^^J%
240   \else
```

`\input@path` has been defined in `texsys.cfg`.

```
241     have different search paths.^^J%
242     LaTeX will use the path specified by \noexpand\input@path:^^J%
243   \fi
244 }
```

(End definition for `\input@path`.)

6 Filename Parsing

`\filename@parse` Split a filename into its components.

```
245 \ifx\filename@parse\@undefined
246   \def\reserved@a{./}\ifx\@currdir\reserved@a
```

`\filename@parse` was not specified in `texsys.cfg`, but `\@currdir` looks like UNIX...

```
247   \typeout{^^JDefining UNIX/DOS style filename parser.^^J}
248   \def\filename@parse#1{%
249     \let\filename@area\empty
250     \expandafter\filename@path#1/\}}
```

Search for the last /.

```
251   \def\filename@path#1/#2\\{%
252     \ifx\\#2\\%
253       \def\reserved@a{\filename@simple#1.\}%
254     \else
255       \edef\filename@area{\filename@area#1\}%
256       \def\reserved@a{\filename@path#2\}%
257     \fi
258   \reserved@a}
259 \else\def\reserved@a{}{}\ifx\@currdir\reserved@a
```

```

\filename@parse was not specified in texsys.cfg, but \currdir looks like VMS...
260   \typeout{^^JDefining VMS style filename parser.^^J}
261   \def\filename@parse#1{%
262     \let\filename@area\empty
263     \expandafter\filename@path#1\\}
Search for the last ].
264   \def\filename@path#1]#2\\{%
265     \ifx\\#2\\%
266       \def\reserved@a{\filename@simple#1.\\}%
267     \else
268       \edef\filename@area{\filename@area#1}%
269       \def\reserved@a{\filename@path#2\\}%
270     \fi
271     \reserved@a}
272   \else\def\reserved@a:{}\ifx\currdir\reserved@a
\filename@parse was not specified in texsys.cfg, but \currdir looks like Macintosh...
273   \typeout{^^JDefining Mac style filename parser.^^J}
274   \def\filename@parse#1{%
275     \let\filename@area\empty
276     \expandafter\filename@path#1:\\}
Search for the last :.
277   \def\filename@path#1:#2\\{%
278     \ifx\\#2\\%
279       \def\reserved@a{\filename@simple#1.\\}%
280     \else
281       \edef\filename@area{\filename@area#1:}%
282       \def\reserved@a{\filename@path#2\\}%
283     \fi
284     \reserved@a}
285   \else
\filename@parse was not specified in texsys.cfg. So just make a simple parser that
always sets \filename@area to empty.
286   \typeout{^^JDefining generic filename parser.^^J}
287   \def\filename@parse#1{%
288     \let\filename@area\empty
289     \expandafter\filename@simple#1.\\}
290   \fi\fi\fi
\filename@simple is used by all three versions. Finally we can split off the exten-
sion.
291   </dircheck>
292   (*dircheck, latexrelease)
293   (<latexrelease>) \IncludeInRelease{2019/10/01}{\filename@simple}
294   (<latexrelease>)                                     {Final dot for extension}%
295   \def\filename@simple#1.#2\\{%
296     \ifx\\#2\\%
297       \let\filename@ext\relax
298       \edef\filename@base{#1}%
299     \else

```

```

300      \filename@dots{\#1}\#2\\%
301  \fi}
302 \def\filename@dots{\#1\#2.\#3\\%
303   \ifx\#3\\%
304     \def\filename@ext{\#2}%
305     \edef\filename@base{\#1}%
306   \else
307     \filename@dots{\#1.\#2}\#3\\%
308   \fi}
309 \EndIncludeInRelease
310 \IncludeInRelease{0000/00/00}{\filename@simple}
311 \Final dot for extension}%
312 \def\filename@simple{\#1.\#2\\%
313 \ifx\#2\\%
314   \let\filename@ext\relax
315 \else
316   \edef\filename@ext{\filename@dot\#2\\%}
317 \fi
318 \edef\filename@base{\#1}%
319 \EndIncludeInRelease
320 \dircheck, latexrelease)
321 {*dircheck}

```

Remove a final dot, added earlier.

```

322 \def\filename@dot{\#1.\\\#1}
323 \else

```

Otherwise, `\filename@parse` was specified in `texsys.cfg`.

```

324 \typeout{^^J^^J%
325   \noexpand\filename@parse was defined in texsys.cfg:^^J%
326   \expandafter\strip@prefix\meaning\filename@parse.^^J%
327 }
328 \fi

```

(*End definition for `\filename@parse`.*)

7 TeX Versions

`\@TeXversion` TeX versions older than 3.141 require `\@TeXversion` to be set. This can be determined automatically due to a trick suggested by Bernd Raichle. (Actually this will not always get the correct version number, eg TeX3.14 would be detected as TeX3, but L^AT_EX only needs to take account of TeX's older than 3, or between 3 and 3.14.

```

329 \ifx\@TeXversion\undefined
330   \ifx\@undefined\inputlineno
331     \def\@TeXversion{2}
332   \else
333     {\catcode`\\=active
334      \def\reserved@a{\if#1\string^3\fi}
335      \edef\reserved@a{\expandafter\reserved@a\string^3\@c}
336      \ifx\reserved@a\empty\else\gdef\@TeXversion{3}\fi}
337   \fi
338 \fi

```

(End definition for \c@TeXversion.)

```
339 %% ---- END temporary definitions for bootstrapping ----
340 </dircheck>
```

8 ltxcheck.tex

After the format has been made, and article.cls moved with the other files to the ‘standard input directory’ as specified in `install.txt`, the format may be checked by running the file `ltxcheck.tex`.

File b

ltplain.dtx

1 Plain T_EX

L^AT_EX includes almost all of the functionality of Knuth's original 'Basic Macros'. That is, the plain T_EX format described in Appendix B of the T_EXBook. However, some of the user commands are not much use so, in order to save memory, we may remove them from the kernel into a package. Here is a list of the commands that may be removed (PROBABLY NOT COMPLETE).

```
\magstep     \magstephalf  
\mathhexbox  
\vglue      \vgl@  
\hglue      \hgl@
```

This file is by now very small as most of it has been moved to more appropriate kernel files: it may disappear completely one day.

L^AT_EX font definitions are done using NFSS2 so none of PLAIN's font definitions are in L^AT_EX.

L^AT_EX has its own tabbing environment, so PLAIN's is disabled.

L^AT_EX uses its own output routine, so most of the plain one was removed.

```
1  {*2ekernel}  
2  \catcode`{\=1 % left brace is begin-group character  
3  \catcode`}=\=2 % right brace is end-group character  
4  \catcode`$=\=3 % dollar sign is math shift  
5  \catcode`&=\=4 % ampersand is alignment tab  
6  \catcode`#=\=6 % hash mark is macro parameter character  
7  \catcode`^=\=7 % circumflex and uparrow are for superscripts  
8  \catcode`_=\=8 % underline and downarrow are for subscripts  
9  \catcode`^^I=\=10 % ascii tab is a blank space  
10 \chardef\active=13 \catcode`~=\active % tilde is active  
11 \catcode`^^L=\active \def^^L{\par}% ascii form-feed is \par  
12 \message{catcodes,}
```

We had to define the \catcodes right away, before the message line, since \message uses the { and } characters. When INITEX (the T_EX initializer) starts up, it has defined the following \catcode values:

```
\catcode`^^@=\=9 % ascii null is ignored  
\catcode`^^M=\=5 % ascii return is end-line  
\catcode`\\=\=0 % backslash is TeX escape character  
\catcode`%=\=14 % percent sign is comment character  
\catcode` =\=10 % ascii space is blank space  
\catcode`^^?=\=15 % ascii delete is invalid  
\catcode`A=\=11 ... \catcode`Z=\=11 % uppercase letters  
\catcode`a=\=11 ... \catcode`z=\=11 % lowercase letters  
all others are type 12 (other)
```

Here is a list of the characters that have been specially catcoded:

```
13 \def\dospecials{\do\ \do\\\do\{\do\}\do\$\\do\&%  
14 \do\#\do\^\do\_\\do\%\do\~}
```

(not counting ascii null, tab, linefeed, formfeed, return, delete) Each symbol in the list is preceded by , which can be defined if you want to do something to every item in the list.

We make @ signs act like letters, temporarily, to avoid conflict between user names and internal control sequences of plain format.

15 \catcode`@=11

To make the plain macros more efficient in time and space, several constant values are declared here as control sequences. If they were changed, anything could happen; so they are private symbols.

\@ne Small constants are defined using \chardef.

\tw@ 16 \chardef\@ne=1

\thr@@ 17 \chardef\tw@=2

\sixt@@n 18 \chardef\thr@@=3

\@cclv 19 \chardef\sixt@@n=16

20 \chardef\@cclv=255

(End definition for \@ne and others.)

\@cclvi Constants above 255 defined using \mathchardef.

\@m 21 \mathchardef\@cclvi=256

\@M 22 \mathchardef\@m=1000

\@MM 23 \mathchardef\@M=10000

24 \mathchardef\@MM=20000

(End definition for \@cclvi and others.)

Allocation of registers

Here are macros for the automatic allocation of \count, \box, \dimen, \skip, \muskip, and \toks registers, as well as \read and \write stream numbers, \fam codes, \language codes, and \insert numbers.

25 \message{registers,}

When a register is used only temporarily, it need not be allocated; grouping can be used, making the value previously in the register return after the close of the group. The main use of these macros is for registers that are defined by one macro and used by others, possibly at different nesting levels. All such registers should be defined through these macros; otherwise conflicts may occur, especially when two or more macro packages are being used at the same time.

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

The following counters are reserved:

0 to 9 page numbering

10 count allocation

11 dimen allocation

12 skip allocation

13 muskip allocation

14 box allocation

15 toks allocation

16 read file allocation

17 write file allocation

18 math family allocation

19 language allocation

20 insert allocation

21 the most recently allocated number

22 constant -1
End of historical L^AT_EX 2.09 comments.

New counters are allocated starting with 23, 24, etc. Other registers are allocated starting with 10. This leaves 0 through 9 for the user to play with safely, except that counts 0 to 9 are considered to be the page and subpage numbers (since they are displayed during output). In this scheme, \count 10 always contains the number of the highest-numbered counter that has been allocated, \count 14 the highest-numbered box, etc. Inserts are given numbers 254, 253, etc., since they require a \count, \dimen, \skip, and \box all with the same number; \count 20 contains the lowest-numbered insert that has been allocated. Of course, \box255 is reserved for \output; \count255, \dimen255, and \skip255 can be used freely.

It is recommended that macro designers always use \global assignments with respect to registers numbered

1, 3, 5, 7, 9,

and always non-\global assignments with respect to registers

0, 2, 4, 6, 8, 255.

This will prevent “save stack buildup” that might otherwise occur.

```
26 \count10=22 % allocates \count registers 23, 24, ...
27 \count11=9 % allocates \dimen registers 10, 11, ...
28 \count12=9 % allocates \skip registers 10, 11, ...
29 \count13=9 % allocates \muskip registers 10, 11, ...
30 \count14=9 % allocates \box registers 10, 11, ...
31 \count15=9 % allocates \toks registers 10, 11, ...
32 \count16=-1 % allocates input streams 0, 1, ...
33 \count17=-1 % allocates output streams 0, 1, ...
34 \count18=3 % allocates math families 4, 5, ...
35 \count19=0 % allocates \language codes 1, 2, ...
36 \count20=255 % allocates insertions 254, 253, ...
```

\insc@unt The insertion counter and most recent allocation.
\allocationnumber

```
37 \countdef\insc@unt=20
38 \countdef\allocationnumber=21
```

(*End definition for \insc@unt and \allocationnumber.*)

\m@ne The constant -1.

```
39 \countdef\m@ne=22 \m@ne=-1
```

(*End definition for \m@ne.*)

\wlog Write on log file (only)

```
40 \def\wlog{\immediate\write\m@ne}
```

(*End definition for \wlog.*)

\count@ Here are abbreviations for the names of scratch registers that don't need to be allocated.
\dimen@
\dimen@i
\dimen@ii
\skip@
\toks@

```
41 \countdef\count@=255
```

```
42 \dimendef\dimen@=0
```

```
43 \dimendef\dimen@i=1 % global only
```

```
44 \dimendef\dimen@ii=2
```

```
45 \skipdef\skip@=0
```

```
46 \toksdef\toks@=0
```

(End definition for `\count@` and others.)

```
\newcount Now, we define \newcount, \newbox, etc. so that you can say \newcount\foo and \foo  
\newdimen will be defined (with \countdef) to be the next counter.  
\newskip To find out which counter \foo is, you can look at \allocationnumber.  
\newmuskip Since there's no \boxdef command, \chardef is used to define a \newbox,  
\newbox, \newinsert, \newfam, and so on.  
\newtoks LATEX change: remove \outer from \newcount and \newdimen (FMi) This is nec-  
\newread essary to use \newcount inside \if... later on. Also remove from \newskip, \newbox  
\newwrite and \newfam (DPC) to save later redefinition.  
\newfam  
\newlanguage 47 </2ekernel>  
48 {*2ekernel | latexrelease}  
49 <| latexrelease> \IncludeInRelease{2015/01/01}%  
50 <| latexrelease> {\newcount}{Extended Allocation}%  
51 \def\newcount {\e@alloc\count \countdef {\count10}\insc@unt\flo@at@count}  
52 \def\newdimen {\e@alloc\dimen \dimendef {\count11}\insc@unt\flo@at@count}  
53 \def\newskip {\e@alloc\skip \skipdef {\count12}\insc@unt\flo@at@count}  
54 \def\newmuskip  
55 {\e@alloc\muskip\muskipdef{\count13}\m@ne\e@alloc@top}
```

For compatibility use \chardef in the classical range.

```
56 \def\newbox {\e@alloc\box  
57 {\ifnum\allocationnumber<\@ccclvi  
58 \expandafter\chardef  
59 \else  
60 \expandafter\@alloc@chardef  
61 \fi}  
62 {\count14}\insc@unt\flo@at@count}  
63 \def\newtoks {\e@alloc\toks \toksdef{\count15}\m@ne\@alloc@top}  
64 \def\newread {\e@alloc\read \chardef{\count16}\m@ne\sixt@@n}  
Skip \write18 due to its traditional use as a shell-escape.  
65 \ifx\directlua\@undefined  
66 \def\newwrite {\e@alloc\write \chardef{\count17}\m@ne\sixt@@n}  
67 \else  
68 \def\newwrite {\e@alloc\write  
69 {\ifnum\allocationnumber=18  
70 \advance\count17\@ne  
71 \allocationnumber\count17 %  
72 \fi  
73 \global\chardef} %  
74 {\count17} %  
75 \m@ne  
76 {128}}  
77 \fi  
78 \def\new@mathgroup  
79 {\e@alloc\mathgroup\chardef{\count18}\m@ne\@mathgroup@top}  
80 \let\newfam\new@mathgroup  
81 \ifx\directlua\@undefined  
82 \def\newlanguage {\e@alloc\language \chardef{\count19}\m@ne\@ccclvi}  
83 \else  
84 \def\newlanguage {\e@alloc\language \chardef{\count19}\m@ne{16384}}
```

```

85 \fi
86 </2ekernel | latexrelease>
87 <latexrelease>\EndIncludeInRelease
88 <latexrelease>\IncludeInRelease{0000/00/00}%
89 <latexrelease> {\newcount}{Extended Allocation}%
90 <latexrelease>\def\newcount{\alloc@0\count\countdef\insc@unt}
91 <latexrelease>\def\newdimen{\alloc@1\dimen\dimendef\insc@unt}
92 <latexrelease>\def\newskip{\alloc@2\skip\skipdef\insc@unt}
93 <latexrelease>\def\newmuskip{\alloc@3\muskip\muskipdef\ccclvi}
94 <latexrelease>\def\newbox{\alloc@4\box\chardef\insc@unt}
95 <latexrelease>\def\newtoks{\alloc@5\toks\toksdef\ccclvi}
96 <latexrelease>\def\newread{\alloc@6\read\chardef\sixt@n}
97 <latexrelease>\def\newwrite{\alloc@7\write\chardef\sixt@n}
98 <latexrelease>\def\new@mathgroup{\alloc@8\fam\chardef\sixt@n}
99 <latexrelease>\def\newlanguage{\alloc@9\language\chardef\ccclvi}
100 <latexrelease>\let\newfam\new@mathgroup
101 <latexrelease>\EndIncludeInRelease

```

(End definition for `\newcount` and others.)

`\e@alloc@chardef` The upper limit of extended registers, which leaves this number (eg `\dimen32767`) always unallocated by these macros. cf traditional `\dimen255`.

```

102 <*2ekernel | latexrelease>
103 <latexrelease>\IncludeInRelease{2015/01/01}%
104 <latexrelease> {\e@alloc@chardef}{Extended Allocation}%
105 \ifx\directlua\undefined
106   \ifx\widowpenalties\undefined

```

classic tex has 2^8 registers.

```

107   \mathchardef\e@alloc@top=255
108   \let\@alloc@chardef\mathchardef
109 \else

```

etex and xetex have 2^{15} registers.

```

110   \mathchardef\@alloc@top=32767
111   \let\@alloc@chardef\mathchardef
112   \fi
113 \else

```

luatex has 2^{16} registers.

```

114   \chardef\@alloc@top=65535
115   \let\@alloc@chardef\mathchardef
116 \fi
117 </2ekernel | latexrelease>
118 <latexrelease>\EndIncludeInRelease
119 <latexrelease>\IncludeInRelease{0000/00/00}%
120 <latexrelease> {\e@alloc@chardef}{Extended Allocation}%
121 <latexrelease>\let\@alloc@top\undefined
122 <latexrelease>\let\@alloc@chardef\undefined
123 <latexrelease>\EndIncludeInRelease

```

(End definition for `\e@alloc@chardef` and `\e@alloc@top`.)

<code>\e@mathgroup@top</code>	The upper limit of extended math groups (<code>\fam</code>) 16 in classic TeX and e-TeX, but 256 in Unicode TeX variants. <pre> 124 {*2ekernel latexrelease} 125 <tex>\IncludeInRelease{2015/01/01}% 126 <tex>\fam{\e@mathgroup@top}{Extended Allocation}% 127 \ifx\Umathcode\undefined </pre> <p>classic and e tex have 16 fam (0–15).</p> <pre> 128 \chardef\mathgroup@top=16 129 \else </pre> <p>xetex and luatex have 256 fam (0–255).</p> <pre> 130 \chardef\mathgroup@top=256 131 \fi 132 </2ekernel latexrelease> 133 <tex>\EndIncludeInRelease 134 <tex>\IncludeInRelease{0000/00/00}% 135 <tex>\fam{\e@mathgroup@top}{Extended Allocation}% 136 <tex>\let\mathgroup@top\undefined 137 <tex>\EndIncludeInRelease </pre> <p>(End definition for <code>\e@mathgroup@top</code>.)</p>
<code>\e@alloc</code>	A modified version of <code>\alloc@</code> that takes the count register rather than just the final digit of its number (assuming <code>\count1x</code>). It also has an extra argument to give the top of the extended range. <pre> #1 #2 #3 #4 #5 #6 \@alloc type defcmd current top extended-top newname </pre> <p>Note that if just a single allocation range is required (not omitting a range up to 255 for inserts) then –1 should be used for the first upper bound argument, #4.</p> <pre> 138 {*2ekernel latexrelease} 139 <tex>\IncludeInRelease{2015/01/01}{\e@alloc}{Extended Allocation}% 140 \def\@alloc#1#2#3#4#5#6{% 141 \global\advance#3\@ne 142 \e@ch@ck{#3}{#4}{#5}{#1}% 143 \allocationnumber#3\relax 144 \global#2#6\allocationnumber 145 \wlog{\string#6=\string#1\the\allocationnumber}% 146 </2ekernel latexrelease> 147 <tex>\EndIncludeInRelease 148 <tex>\IncludeInRelease{0000/00/00}{\e@alloc}{Extended Allocation}% 149 <tex>\let\@alloc\undefined 150 <tex>\EndIncludeInRelease 151 </2ekernel> </pre> <p>(End definition for <code>\e@alloc</code>.)</p>
<code>\e@ch@ck</code>	Extended check command. If the first range is exceeded, bump to 256 (or 266 for counts) and try again, testing the extended range.

```

Allocate matching registers from the top of the extended range and add to \c@freelist.

\extrafloats 152  {/2ekernel}
              {*2ekernel | latexrelease}
              {latexrelease\IncludeInRelease{2015/10/01}
               {latexrelease}{\e@ch@ck}{Extended Allocation (checking)}}

156  \gdef\c@ch@ck#1#2#3#4{%
157    \ifnum#1<#2\else

```

If we've reached the classical top limit, bump to 256 or 266 for counts (count 256–265 are reserved by the allocation system).

```

158    \ifnum#1=#2\relax
159      \global\c@cclvi
160      \ifx\c@count\c@global\advance\c@count 10 \fi
161    \fi

```

Check we are below the extended limit.

```

162    \ifnum#1<#3\relax
163    \else
164      \errmessage{No room for a new \string#4}%
165    \fi
166  \fi}%
167  {latexrelease\EndIncludeInRelease
168  {latexrelease\IncludeInRelease{2015/01/01}%
169  {latexrelease}{\e@ch@ck}{Extended Allocation (checking)}}
170  {latexrelease\gdef\c@ch@ck#1#2#3#4{%
171    \ifnum#1<#2\else
172      \ifnum#1=#2\relax
173        \c@cclvi
174        \ifx\c@count\c@global\advance\c@count 10 \fi
175      \fi
176      \ifnum#1<#3\relax
177      \else
178        \errmessage{No room for a new #4}%
179      \fi
180    \fi}%
181  \EndIncludeInRelease
182  {latexrelease\IncludeInRelease{0000/00/00}%
183  {latexrelease}{\e@ch@ck}{Extended Allocation (checking)}}
184  {latexrelease\let\c@ch@ck\c@undefined
185  \EndIncludeInRelease
186  {latexrelease\IncludeInRelease{2015/01/01}%
187  {latexrelease}{\extrafloats}{Extra floats}%
188 \let\c@float\c@count\c@alloc@\top

```

```

\extrafloats 189 \ifx\c@numexpr\c@undefined

```

In classic TeX use \newinsert to allocate float boxes.

```

190 \def\extrafloats#1{%
191 \c@count@#1\relax
192 \ifnum\c@count@>\z@
193 \newinsert\c@reserved@a
194 \c@global\expandafter\chardef

```

```

195          \csname bx@\the\allocationnumber\endcsname\allocationnumber
196  \@cons\@freelist{\csname bx@\the\allocationnumber\endcsname}%
197  \advance\count@\m@ne
198  \expandafter\extrafloats
199  \expandafter\count@
200 \fi
201 }%
202 \else

```

In e-tex take float boxes from the top of the extended range.

```

203 \def\extrafloats#1{%
204 \ifnum#1>\z@
205 \count@\numexpr\float@count-1\relax
206 \ch@ck0\count@\count
207 \ch@ck1\count@\dimen
208 \ch@ck2\count@\skip
209 \ch@ck4\count@\box
210 \global\edef@alloc@chardef\float@count\count@
211 \global\expandafter\edef@alloc@chardef
212 \csname bx@\the\float@count\endcsname\float@count
213 \@cons\@freelist{\csname bx@\the\float@count\endcsname}%
214 \expandafter
215 \extrafloats\expandafter{\numexpr#1-1\relax}%
216 \fi}%
217 \fi
218 </2ekernel | latexrelease>
219 <latexrelease>\EndIncludeInRelease
220 <latexrelease>\IncludeInRelease{0000/00/00}%
221 <latexrelease>           {\extrafloats}{Extra floats}%
222 <latexrelease>\let\float@count\undefined
223 <latexrelease>\let\extrafloats\undefined
224 <latexrelease>\EndIncludeInRelease
225 <*2ekernel>

```

(End definition for `\e@ch@ck`, `\extrafloats`, and `\extrafloats`.)

\alloc@ Since `\e@alloc` was added in 2015, `\alloc` has not been used, but was left as some legacy code calls it. However the original definition gives spurious errors once the “classic” registers run out, so it is now defined to call `\e@alloc` internally.

```

226 </2ekernel>
227 <*2ekernel | latexrelease>
228 <latexrelease>\IncludeInRelease{2020/10/01}
229 <latexrelease>           {\alloc@}{emulate alloc@}%
230 \def\alloc@#1#2#3#4{\e@alloc#2#3{\count1#1}#4\float@count}
231 </2ekernel | latexrelease>

232 <latexrelease>\EndIncludeInRelease
233 <latexrelease>\IncludeInRelease{0000/00/00}%
234 <latexrelease>           {\alloc@}{emulate alloc@}%
235 <latexrelease>\def\alloc@#1#2#3#4#5{\global\advance\count1#1\@ne
236 <latexrelease> \ch@ck1#4#2%
237 <latexrelease> \allocationnumber\count1#1%
238 <latexrelease> \global#3#5\allocationnumber
239 <latexrelease> \wlog{string#5=string#2\the\allocationnumber}%

```

```

240  ⟨\latexrelease⟩\EndIncludeInRelease
241  ⟨*2ekernel⟩

(End definition for \alloc@.)
```

\newinsert

```

242  ⟨/2ekernel⟩
243  ⟨*2ekernel | \latexrelease⟩
244  ⟨\latexrelease⟩\IncludeInRelease{2015/10/01}
245  ⟨\latexrelease⟩           {\newinsert}{Extended \newinsert}%
246  \ifx\numexpr\undefined
```

If e-TeX is not available use the original plain TeX definition of \newinsert.

```

247 \def\newinsert#1{\global\advance\insc@unt \m@ne
248   \ch@ck0\insc@unt\count
249   \ch@ck1\insc@unt\dimen
250   \ch@ck2\insc@unt\skip
251   \ch@ck4\insc@unt\box
252   \allocationnumber\insc@unt
253   \global\chardef#1\allocationnumber
254   \wlog{\string#1=\string\insert\the\allocationnumber}%
255 \else
```

The highest register allowed with \insert.

```

256 \ifx\directlua\undefined
257   \chardef\@insert@top255
258 \else
259   \chardef\@insert@top\@alloc@top
260 \fi
```

If the classic registers are exhausted, take an insert from the free float list and use \extrafloats to add a new float to that list.

```

261 \def\newinsert#1{%
262   \tempswafalse
263   \global\advance\insc@unt\m@ne
264   \ifnum\count10<\insc@unt
265   \ifnum\count11<\insc@unt
266   \ifnum\count12<\insc@unt
267   \ifnum\count14<\insc@unt
268     \tempswatrue
269   \fi\fi\fi\fi
270   \if@tempswa
271     \allocationnumber\insc@unt
272   \else
273     \global\advance\insc@unt\@ne
274     \extrafloats\@ne
275     \next@\currbox\@freelist
276     {\ifnum\currbox<\@insert@top
277       \allocationnumber\currbox
278     \else
279       \ch@ck0\m@ne\insert
280     \fi}%
281     {\ch@ck0\m@ne\insert}%
282 \fi
```

```

283 \global\chardef#1\allocationnumber
284 \wlog{\string#1=\string\insert\the\allocationnumber}%
285 }

286 \fi
287 </2ekernel | latexrelease>

288 <latexrelease>\EndIncludeInRelease
289 <latexrelease>\IncludeInRelease{0000/00/00}%
290 <latexrelease>          {\newinsert}{Extended \newinsert}%
291 <latexrelease>\let\@insert@top\@undefined
292 <latexrelease>\def\newinsert#1{\global\advance\insc@unt \m@ne
293 <latexrelease> \ch@ck0\insc@unt\count
294 <latexrelease> \ch@ck1\insc@unt\dimen
295 <latexrelease> \ch@ck2\insc@unt\skip
296 <latexrelease> \ch@ck4\insc@unt\box
297 <latexrelease> \allocationnumber\insc@unt
298 <latexrelease> \global\chardef#1\allocationnumber
299 <latexrelease> \wlog{\string#1=\string\insert\the\allocationnumber}%
300 <latexrelease>\EndIncludeInRelease
301 <2ekernel>

```

(End definition for `\newinsert`.)

```
\ch@ck
302 \gdef\ch@ck#1#2#3{%
303   \ifnum\count1#1<#2\else
304     \errmessage{No room for a new #3}%
305   \fi}

```

(End definition for `\ch@ck`.)

```
\newhelp
306 \def\newhelp#1#2{\newtoks#1#1\expandafter{\csname#2\endcsname}}

```

(End definition for `\newhelp`.)

`\@inputcheck` Allocate read stream for testing and output stream that is never open an thus writes to the terminal.

```
307 \newread\@inputcheck
308 \newwrite\@unused
```

(End definition for `\@inputcheck` and `\@unused`.)

`\maxdimen` Here are some examples of allocation.

```
\hideskip
309 \newdimen\maxdimen \maxdimen=16383.99999pt % the largest legal <dimen>
310 \newskip\hideskip \hideskip=-1000pt plus 1fill % negative but can grow
```

(End definition for `\maxdimen` and `\hideskip`.)

```
\p@
\z@
311 \newdimen\p@ \p@=1pt % this saves macro space and time
\z@skip
312 \newdimen\z@ \z@=0pt % can be used both for Opt and 0
\voidb@x
313 \newskip\z@skip \z@skip=Opt plus0pt minusOpt
314 \newbox\voidb@x % permanently void box register
```

(End definition for `\p@` and others.)

Assign initial values to TeX's parameters

315 `\message{parameters,}`

All of TeX's numeric parameters are listed here, but the code is commented out if no special value needs to be set. INITEX makes all parameters zero except where noted.

Historical LaTeX 2.09 comments (not necessarily accurate any more):

```
316 \pretolerance=100
317 \tolerance=200 % INITEX sets this to 10000
318 \hbadness=1000
319 \vbadness=1000
320 \linepenalty=10
321 \hyphenpenalty=50
322 \exhyphenpenalty=50
323 \binoppenalty=700
324 \relpenalty=500
325 \clubpenalty=150
326 \widowpenalty=150
327 \displaywidowpenalty=50
328 \brokenpenalty=100
329 \predisplaypenalty=10000
\postdisplaypenalty=0
\interlinepenalty=0
\floatingpenalty=0, set during \insert
\outputpenalty=0, set before TeX enters \output
330 \doublehyphendemerits=10000
331 \finalhyphendemerits=5000
332 \adjdemerits=10000
333 % \looseness=0, cleared by TeX after each paragraph
334 % \pausing=0
335 % \holdinginserts=0
336 % \tracingonline=0
337 % \tracingmacros=0
338 % \tracingstats=0
339 % \tracingparagraphs=0
340 % \tracingpages=0
341 % \tracingoutput=0
```

In the past `\LaTeX{}` used the default value of `\texttt{1}` for `\cs{tracinglostchars}` because this was the best it could do. This way one would at least get a warning in the `\texttt{.log}` file. e-`\TeX{}` improved on that and supported a value of `\texttt{2}` to show the warning on the terminal, so we could have changed the default when we made the e-`\TeX{}` extensions required—however, we overlooked that opportunity.

In 2021 this parameter was improved on again and now also accepts the value `\texttt{3}`

(error on the terminal). This made us realize that we should change the default. Using `\texttt{3}` would really be the best, but for compatibility reasons we only use `\texttt{2}`.

`\changes{v2.3g}{2021/07/16}{Use 2 as default value for \cs{tracinglostchars}}`

342 `\tracinglostchars=2`

```

343 % \tracingcommands=0
344 % \tracingrestores=0

\begin{macro}{\tracingstacklevels}
  For Lua\TeX, the \cs{\tracingstacklevels} functionality was
  implemented as a callback, so here we just define the count register
  to hold the value of the parameter.

345 \langle/2ekernel\rangle
346 \langle*2ekernel | latexrelease\rangle
347 \langle latexrelease \rangle \IncludeInRelease{2021/06/01}{\tracingstacklevels}%
348 \langle latexrelease \rangle \tracingstacklevels\%
349 \ifx\directlua\undefined
350   % \tracingstacklevels=0 % added in 2021
351 \else
352   \newcount\tracingstacklevels
353   % Code for \tracingstacklevels defined in ltfinal.dtx
354 \fi
355 \langle latexrelease \rangle \EndIncludeInRelease
356 \langle latexrelease \rangle
357 \langle latexrelease \rangle \IncludeInRelease{0000/00/00}{\tracingstacklevels}%
358 \langle latexrelease \rangle \tracingstacklevels\%
359 \langle latexrelease \rangle \ifx\directlua\undefined
360 \langle latexrelease \rangle \else
361 \langle latexrelease \rangle \let\tracingstacklevels\undefined
362 \langle latexrelease \rangle \fi
363 \langle latexrelease \rangle \EndIncludeInRelease
364 \langle/2ekernel | latexrelease\rangle
365 \langle*2ekernel\rangle

\end{macro}

\language=0

366 \uchyph=1

\lefthyphenmin=2 \righthyphenmin=3 set below
\globaldefs=0
\maxdeadcycles=25 % INITEX does this
\hangafter=1 % INITEX does this, also TeX after each paragraph
\fam=0
\mag=1000 % INITEX does this
\escapechar='\\ % INITEX does this

367 \defaulthyphenchar='-
368 \defaultskewchar=-1

\endlinechar='\^M % INITEX does this
\newlinechar=-1 \LaTeX\ sets this in ltdefns.dtx.

369 \delimiterfactor=901

```

```
\time=now % TeX does this at beginning of job
\day=now % TeX does this at beginning of job
\month=now % TeX does this at beginning of job
\year=now % TeX does this at beginning of job
```

End of historical L^AT_EX 2.09 comments.

In L^AT_EX we don't want box information in the transcript unless we do a full tracing.

```
370 \showboxbreadth=-1
371 \showboxdepth=-1
372 \errorcontextlines=-1
373 \hfuzz=0.1pt
374 \vfuzz=0.1pt
375 \overfullrule=5pt
376 \maxdepth=4pt
377 \splitmaxdepth=\maxdimen
378 \boxmaxdepth=\maxdimen
```

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

\lineskiplimit=0pt, changed by \normalbaselines

```
379 \delimitershortfall=5pt
380 \nulldelimiterspace=1.2pt
381 \scriptspace=0.5pt
\mathsurround=0pt
\predisplaysize=0pt, set before TeX enters $$
\displaywidth=0pt, set before TeX enters $$
\displayindent=0pt, set before TeX enters $$
382 \parindent=20pt
\hangindent=0pt, zeroed by TeX after each paragraph
\hoffset=0pt
\voffset=0pt
```

\baselineskip=0pt, changed by \normalbaselines
\lineskip=0pt, changed by \normalbaselines

```
383 \parskip=0pt plus 1pt
384 \abovedisplayskip=12pt plus 3pt minus 9pt
385 \abovedisplayshortskip=0pt plus 3pt
386 \belowdisplayskip=12pt plus 3pt minus 9pt
387 \belowdisplayshortskip=7pt plus 3pt minus 4pt
\leftskip=0pt
\rightskip=0pt
388 \topskip=10pt
389 \splittopskip=10pt
\tabskip=0pt
\spaceskip=0pt
\xspaceskip=0pt
390 \parfillskip=0pt plus 1fil
```

End of historical L^AT_EX 2.09 comments.

\normalbaselineskip
 \ normallineskip
\normallineskiplimit
We also define special registers that function like parameters:
391 \newskip\normalbaselineskip \normalbaselineskip=12pt
392 \newskip\normallineskip \normallineskip=1pt
393 \newdimen\normallineskiplimit \normallineskiplimit=0pt
(*End definition for \normalbaselineskip, \normallineskip, and \normallineskiplimit.*)

\interfootlinepenalty
394 \newcount\interfootnotelinepenalty \interfootnotelinepenalty=100
(*End definition for \interfootlinepenalty.*)
Definitions for preloaded fonts

\magstephalf
 \magstep
395 \def\magstephalf{1095 }
396 \def\magstep#1{\ifcase#1 \or 1200\or 1440\or 1728\or
397 2074\or 2488\fi\relax}
(*End definition for \magstephalf and \magstep.*)

Macros for setting ordinary text

\frenchspacing
\nonfrenchspacing
398 \def\frenchspacing{\sfcode‘.\.‘\@m \sfcode‘?\@m \sfcode‘!\@m
399 \sfcode‘:\@m \sfcode‘\;‘\@m \sfcode‘\,\@m}
400 \def\nonfrenchspacing{\sfcode‘.\.3000\sfcode‘?3000\sfcode‘!\!3000%
401 \sfcode‘:\!2000\sfcode‘\;‘1500\sfcode‘\,,‘1250 }
(*End definition for \frenchspacing and \nonfrenchspacing.*)

\normalbaselines
402 \def\normalbaselines{\lineskip\normallineskip
403 \baselineskip\normalbaselineskip \lineskiplimit\normallineskiplimit}
(*End definition for \normalbaselines.*)

\M Save a bit of space by using \let here.
\I 404 \def\^\M{\ } % control <return> = control <space>
405 \let\^\I\^\M % same for <tab>
(*End definition for \M and \I.*)

\lq
\rq
406 \def\lq{‘}
407 \def\rq{’}
(*End definition for \lq and \rq.*)

\lbrack
\rbrack
408 \def\lbrack{[]}
409 \def\rbrack{} }
(*End definition for \lbrack and \rbrack.*)

\aa These are not from plain.tex but they are similar to other commands found here and
\AA nowhere else, being alternate input forms for characters.

```
410 \def \aa {\r a}
411 \def \AA {\r A}
```

(End definition for \aa and \AA.)

\endgraf

```
\endline 412 \let\endgraf=\par
413 \let\endline=\cr
```

(End definition for \endgraf and \endline. These functions are documented on page 296.)

\space

```
414 \def\space{ }
```

(End definition for \space.)

\empty This probably ought to go altogether, but let it to the L^AT_EX version to save space.

```
415 \let\empty\@empty
```

(End definition for \empty.)

\null

```
416 \def\null{\hbox{}}
```

(End definition for \null.)

\bgroup

```
\egroup 417 \let\bgroup={
418 \let\egroup=}
```

(End definition for \bgroup and \egroup.)

\obeylines In \obeylines, we say \let^{^^M}=\obeyedline instead of \def^{^^M}{\obeyedline} since this allows, for example, \let\obeyedline=\cr \obeylines \halign{....}

This is essentially a plain T_EX trick and in its original version where you had to use to use \let\par=\cr not really a safe idea in L^AT_EX. If anybody used this trick this now breaks (and one needs to use \obeyedline instead).

```
419 </2ekernel>
420 <*2ekernel | latexrelease>
421 <latexrelease>\IncludeInRelease{2022/06/01}{\obeylines}%
422 <latexrelease> {Add a redirection to obeylines and obeyspaces}%
```

If the active ^{^^M} escapes, e.g. into a \write (which is effectively in a different context) we don't want the definition from \obeylines but rather a simple \par (in fact even the primitive one, not the L^AT_EX version \para_end: which is only defined later).

```
423 \begingroup
424 \catcode`^^M=\active % these lines must end with %
425 \gdef\obeylines{\catcode`^^M\active%
426 \let^^M\obeyedline%
```

The next line ending the definition is rather curious and it took me awhile to understand why rollback fails. The problem is the following: if `\textrlease` is used, then blocks of `\IncludeInRelease ... \EndIncludeInRelease` are bypassed at high speed by grabbing each as a delimited argument. However, in that case `^M` is seen not as code but as line ending characters and in that mode `TEX` discards everything from that point onwards to the real end of the line so it works like a comment — pretty strange really (and I think due to the fact the the original pascal compiler could have some garbage showing up after the normal line ending character). Thus we really have to make sure that any closing braces is not one the same line as an `^M`, because otherwise it would get dropped and we end with unbalanced braces and never see the `\EndIncludeInRelease` — weird. In other places it doesn't matter because we aren't using the incomplete result.

```
427  }%
428  \global\let^M\par % this is in case ^M appears in a \write
429  \endgroup
```

`\obeyedline` The `\obeyedline` expands by default to `\par` with whatever definition `\par` has when it is executed. It can, however, be redefined (before calling `\obeylines!`) to achieve some special effects. If you want to alter its definition when already in the scope of `\obeylines`, it has no effect (because `\let` is used above). In that case simply make another call to `\obeylines` immediately. As you are in a restricted scope all that happens is that your redefinition is applied.

For the default definition we have to use `\def` not `\let` because the meaning of `\par` can change and we want to use the one that is current when `\obeylines` act.

There is a small subtlety here: in an `\edef` the active `^M` stayed put (because it was equal to the primitive `\par`), now `\obeyedline` expands and you get what it contains, i.e., in that case `\par`, into the `\edef` or `\mark` unless we use `\protected` on it.

```
430 \protected\gdef\obeyedline{\par}
```

The definition of `\obeyspaces` is changed in the same way and now executes `\obeyedspace` for each active space.

```
\obeyedspace
431 \global\let\obeyedspace\space
432 \begingroup
433 \catcode`\ =\active%
434 \gdef\obeyspaces{\catcode`\ \active\let =\obeyedspace}%

```

An active space elsewhere generates `\space` by default (for example in a `\write`).

```
435 \global\let =\space%
436 \endgroup
437 {/2ekernel | \textrlease}
438 {|\textrlease}\EndIncludeInRelease
439 {|\textrlease}\IncludeInRelease{0000/00/00}{\obeylines}%
440 {|\textrlease}    {Add a redirection to obeylines and obeyspaces}%
441 {|\textrlease}
```

From 2019 onwards the commands are made robust (somewhat later in the kernel sources). So if we roll back they are robust, so when redefining them we have to get rid of the robust payload first. Otherwise that is seen by the later rollback below, which then installs a fragile version of the new definition on top of the one we roll back to here, sigh. `\kernel@make@fragile` also changes its definition (later own) so this is done directly.

```
442 {|\textrlease}\expandafter\let\csname obeylines \endcsname\@undefined
```

```

443 <|latexrelease>\expandafter\let\csname obeyspace \endcsname\@undefined
444 <|latexrelease>
445 <|latexrelease>\begingroup
446 <|latexrelease>\catcode`^\^M=\active % these lines must end with %
447 <|latexrelease> \gdef\obeylines{\catcode`^\^M\active \let^\^M\par %

```

Closing brace on a separate line (see comment above).

```
448 <|latexrelease> }%
```

Another pitfall: if we do a rollback `\par` is no longer the primitive, so the roll back definition needs `\let` to what is now the primitive.

```

449 <|latexrelease> \global\let^\^M\RawParEnd % this is in case `^\^M appears in a \write
450 <|latexrelease>\endgroup
451 <|latexrelease>\def\obeyspaces{\catcode`\ \active}
452 <|latexrelease>
453 <|latexrelease>\let\obeyedline\@undefined
454 <|latexrelease>\let\obeyedspace\@undefined
455 <|latexrelease>\EndIncludeInRelease
456 <|*2ekernel>

```

(*End definition for `\obeylines` and others.*)

`\loop` We use Kabelschacht's method of doing loops, see TUB 8#2 (1987). (unless that breaks
`\iterate` something :-). It turned out to need an extra `\relax`: see pr/642 (`\loop` could do one
`\repeat` iteration too much in certain cases).

```

457 \long\def \loop #1\repeat{%
458   \def\iterate{\#1\relax % Extra \relax
459             \expandafter\iterate\fi
460           }%
461   \iterate
462   \let\iterate\relax
463 }

```

This setting of `\repeat` is needed to make `\loop... \if... \repeat` skippable within another `\if....`

```
464 \let\repeat=\fi
```

(*End definition for `\loop`, `\iterate`, and `\repeat`.*)

LATEX defines `\smallskip`, etc. in `ltspc.dtx`.

```

\nointerlineskip
\offinterlineskip 465 \def\nointerlineskip{\prevdepth-\@m\p@}
466 \def\offinterlineskip{\baselineskip-\@m\p@%
467   \lineskip\z@\lineskiplimit\maxdimen}

```

(*End definition for `\nointerlineskip` and `\offinterlineskip`.*)

```

\vgue
\hgue 468 \def\vgue{\afterassignment\vgl@\skip@=}
469 \def\vgl@{\par \dimen@\prevdepth \hrule \height\z@
470   \nobreak\vskip\skip@ \prevdepth\dimen@}
471 \def\hgue{\afterassignment\hgl@\skip@=}
472 \def\hgl@{\leavevmode \count@\spacefactor \vrule \width\z@
473   \nobreak\hskip\skip@ \spacefactor\count@}

```

(End definition for `\vglue` and `\hglue`.)
`\TeX` defines `\sim` in `ltdefns.dtx`.

`\slash` This generates a / acting a bit like – but still allows hyphenation in the word part preceding it (but not after).

```
474 \def\slash{\penalty\exhyphenpenalty}
```

(End definition for `\slash`.)

`\break`

`\nobreak`

```
475 \def\nobreak{\penalty-\@M}
```

`\allowbreak`

```
476 \def\nobreak{\penalty \@M}
```

```
477 \def\allowbreak{\penalty \z@}
```

(End definition for `\break`, `\nobreak`, and `\allowbreak`.)

`\filbreak`

`\goodbreak`

```
478 \def\filbreak{\par\vfil\penalty-200\vfilneg}
```

```
479 \def\goodbreak{\par\penalty-500 }
```

(End definition for `\filbreak` and `\goodbreak`.)

`\eject` Define `\eject` as in plain `\TeX` but define `\supereject` only in the compatibility file.

```
480 \def\eject{\par\break}
```

(End definition for `\eject`.)

`\removelastskip`

```
481 \def\removelastskip{\ifdim\lastskip=\z@\else\vskip-\lastskip\fi}
```

(End definition for `\removelastskip`.)

`\smallbreak`

`\medbreak`

```
482 \def\smallbreak{\par\ifdim\lastskip<\smallskipamount}
```

`\bigbreak`

```
483 \removelastskip\penalty-50\smallskip\fi}
```

```
484 \def\medbreak{\par\ifdim\lastskip<\medskipamount}
```

```
485 \removelastskip\penalty-100\medskip\fi}
```

```
486 \def\bigbreak{\par\ifdim\lastskip<\bigskipamount}
```

```
487 \removelastskip\penalty-200\bigskip\fi}
```

(End definition for `\smallbreak`, `\medbreak`, and `\bigbreak`.)

`\m@th`

```
488 \def\m@th{\mathsurround\z@}
```

(End definition for `\m@th`.)

`\underbar` Due to `\TeX`'s redefinition of `\underline` plain `\TeX`'s `\underbar` can be done in a simpler fashion (but do we need it at all?).

```
489 \def\underbar{\underline{\sbox\tw@{\#1}\dp\tw@\z@\box\tw@}}
```

(End definition for `\underbar`.)

`\strutbox` `\TeX` sets `\strutbox` in `\set@fontsize`.

`\strut`

```
490 \newbox\strutbox
```

```
491 \def\strut{\relax\ifmmode\copy\strutbox\else\unhcopy\strutbox\fi}
```

(End definition for `\strutbox` and `\strut`.)

`\hidewidth` For alignment entries that can stick out.

492 `\def\hidewidth{\hskip\hideskip}`

(End definition for `\hidewidth`.)

`\narrower`

493 `\def\narrower{%`
494 `\advance\leftskip\parindent`
495 `\advance\rightskip\parindent}`

(End definition for `\narrower`.)

`\textrm` defines `\ae` and similar commands elsewhere.

496 `\chardef\%='\%`
497 `\chardef\&='\&`
498 `\chardef\#= '\#`

Most text commands are actually encoding specific and therefore defined later, so commented out or removed from this file.

`\leavevmode` begins a paragraph, if necessary

499 `\def\leavevmode{\unhbox\voidb@x}`

(End definition for `\leavevmode`.)

`\mathhexbox`

500 `\def\mathhexbox#1#2#3{\mbox{$\m@th \mathchar"#1#2#3$}}`

(End definition for `\mathhexbox`.)

`\ialign`

501 `\def\ialign{\everycr{}\tabskip\z@skip\halign} % initialized \halign`

(End definition for `\ialign`.)

`\oalign`

502 `\def\oalign#1{\leavevmode\vtop{\baselineskip\z@skip \lineskip.25ex%`
503 `\ialign{\##\crcr#1\crcr}}}`
504 `\def\o@align{\lineskip\z@\oalign}`
505 `\def\ooalign{\lineskip\z@-\maxdimen \oalign}`

(End definition for `\oalign`, `\o@align`, and `\ooalign`.)

`\sh@ft` The definition of this macro in plain.tex was improved in about 1997; but as a result its usage was changed and its new definition is not appropriate for `\textrm`.

Since the version given here has been in use by `\textrm` for many years it does not seem prudent to remove it now. As far as we can tell it has only been used to define `\b` and `\d` but this cannot be certain.

506 `\def\sh@ft#1{\dimen@.00#1ex\multiply\dimen@\fontdimen1\font`
507 `\kern-.0156\dimen@} % compensate for slant in lowered accents`

(End definition for `\sh@ft`.)

`\ltx@sh@ft` This is the L^AT_EX version of the second incarnation of the plain macro `\sh@ft`, which takes a dimension as its argument. It shifts a pseudo-accent horizontally by an amount proportional to the product of its argument and the slant-per-point (fontdimen 1).

```
508 \def\ltx@sh@ft #1{%
509   \dimen@ #1%
510   \kern \strip@pt
511   \fontdimen1\font \dimen@
512 } % kern by #1 times the current slant
```

(End definition for `\ltx@sh@ft`.)

L^AT_EX change: the text commands such as `\d`, `\b`, `\c`, `\copyright`, `\TeX` are now defined elsewhere.

L^AT_EX change: Make `\t` work in a moving argument. Now defined elsewhere.

`\hrulefill` L^AT_EX change: `\kern\z@` added to end of `\hrulefill` and `\dotfill` to make them work in ‘tabular’ and ‘array’ environments. (Change made 24 July 1987). L^AT_EX change: `\leavevmode` added at beginning of `\dotfill` and `\hrulefill` so that they work as expected in vertical mode.

```
513 \def\hrulefill{\leavevmode\leaders\hrule\hfill\kern\z@}
```

The box in `\dotfill` originally contained (in plain.tex):

`\mkern 1.5mu . \mkern 1.5mu;`

the width of `.44em` differs from this by `.04pt` which is probably an acceptable difference within leaders.

```
514 \def\dotfill{%
515   \leavevmode
516   \cleaders \hb@xt@ .44em{\hss.\hss}\hfill
517   \kern\z@}
```

(End definition for `\hrulefill` and `\dotfill`.)

INITEX sets `\sfcode x=1000` for all `x`, except that `\sfcode'X=999` for uppercase letters. The following changes are needed:

```
518 \sfcode`)=0 \sfcode`'=0 \sfcode`\]=0
```

The `\nonfrenchspacing` macro will make further changes to `\sfcode` values.

Definitions related to output

`\magnification` doesn’t work in L^AT_EX.

```
def\magnification{\afterassignment\m@g\count@}
def\m@g{\mag\count@
\hsize6.5truein\vsiz8.9truein\dimen\footins8truein}
```

`\showoverfull` The following commands are used in debugging:

```
519 \def\showoverfull{\tracingonline@ne}
```

(End definition for `\showoverfull`.)

`\showoutput`

`\loggingoutput`

```
520 \gdef\loggingoutput{\tracingoutput@ne
521   \showboxbreadth\maxdimen\showboxdepth\maxdimen\errorstopmode}
522 \gdef\showoutput{\loggingoutput\showoverfull}
523 </2ekernel>
```

(End definition for `\showoutput` and `\loggingoutput`.)

```

\tracingall
\loggingall
524  <|latexrelease>\IncludeInRelease{2021/06/01}{\loggingall}
525  <|latexrelease>                                {\tracingstacklevels and \tracinglostchars=3}%
526  <*2ekernel | latexrelease>
527  \edef\loggingall{%
528    \tracingstats\tw@%
529    \tracingpages\@ne%
530    \tracinglostchars\thr@@%
531    \tracingparagraphs\@ne%
532    \tracinggroups\@ne%
533    \tracingifs\@ne%
534    \tracingscantokens\@ne%
535    \tracingnesting\@ne%
536    \errorcontextlines\maxdimen%
537    \ifdef\tracingstacklevels \tracingstacklevels\maxdimen \fi%
538    \noexpand \loggingoutput%
539    \tracingmacros\tw@%
540    \tracingcommands\thr@@%
541    \tracingrestores\@ne%
542    \tracingassigns\@ne%
543  }%
544  \def\tracingall{\showoverfull\loggingall}%
545  </2ekernel | latexrelease>
546  <|latexrelease>\EndIncludeInRelease
547  <|latexrelease>
548  <|latexrelease>\IncludeInRelease{2015/01/01}{\loggingall}{etex tracing}%
549  <|latexrelease>\ifx\tracingscantokens\undefined%
550  <|latexrelease>\gdef\loggingall{%
551  <|latexrelease>  \tracingstats\tw@%
552  <|latexrelease>  \tracingpages\@ne%
553  <|latexrelease>  \tracinglostchars\@ne%
554  <|latexrelease>  \tracingparagraphs\@ne%
555  <|latexrelease>  \errorcontextlines\maxdimen%
556  <|latexrelease>  \loggingoutput%
557  <|latexrelease>  \tracingmacros\tw@%
558  <|latexrelease>  \tracingcommands\tw@%
559  <|latexrelease>  \tracingrestores\@ne%
560  <|latexrelease>  }%
561  <|latexrelease>\else%
562  <|latexrelease>\gdef\loggingall{%
563  <|latexrelease>  \tracingstats\tw@%
564  <|latexrelease>  \tracingpages\@ne%
565  <|latexrelease>  \tracinglostchars\tw@%
566  <|latexrelease>  \tracingparagraphs\@ne%
567  <|latexrelease>  \tracinggroups\@ne%
568  <|latexrelease>  \tracingifs\@ne%
569  <|latexrelease>  \tracingscantokens\@ne%
570  <|latexrelease>  \tracingnesting\@ne%
571  <|latexrelease>  \errorcontextlines\maxdimen%
572  <|latexrelease>  \loggingoutput%
573  <|latexrelease>  \tracingmacros\tw@%
574  <|latexrelease>  \tracingcommands\thr@@%
575  <|latexrelease>  \tracingrestores\@ne%
576  <|latexrelease>  \tracingassigns\@ne%

```

```

577  \end{macro}
578  \end{macro}\fi
579  \gdef\tracingall{\showoverfull\loggingall}
580  \EndIncludeInRelease
581  \end{macro}
582  \IncludeInRelease{0000/00/00}{\loggingall\etex tracing}%
583  \gdef\loggingall{\tracingcommands\tw@\tracingstats\tw@
584  \tracingpages\@ne\tracinglostchars\@ne
585  \tracingmacros\tw@\tracingparagraphs\@ne\tracingrestores\@ne
586  \errorcontextlines\maxdimen\loggingoutput}
587  \gdef\tracingall{\loggingall\showoverfull}
588  \EndIncludeInRelease

```

(End definition for \tracingall and \loggingall.)

\tracingnone

```

589  \IncludeInRelease{2015/01/01}{\tracingnone}%
590  \end{macro}                                {turn off etex tracing}%
591  {*2ekernel | latexrelease}
592  \edef\tracingnone{%
593    \tracingassigns\z@%
594    \tracingrestores\z@%
595    \tracingonline\z@%
596    \tracingcommands\z@%
597    \showboxdepth\m@ne%
598    \showboxbreadth\m@ne%
599    \tracingoutput\z@%
600    \errorcontextlines\m@ne%
601    \ifdefined\tracingstacklevels \tracingstacklevels\z@ \fi%
602    \tracingnesting\z@%
603    \tracingscantokens\z@%
604    \tracingifs\z@%
605    \tracinggroups\z@%
606    \tracingparagraphs\z@%
607    \tracingmacros\z@%
608    \tracinglostchars\@ne%
609    \tracingpages\z@%
610    \tracingstats\z@%
611 }%
612 /*2ekernel | latexrelease)
613 \EndIncludeInRelease
614 \end{macro}
615 \IncludeInRelease{2015/01/01}{\tracingnone}%
616 \end{macro}                                {turn off etex tracing}%
617 \ifx\tracingscantokens\@undefined
618 \def\tracingnone{%
619   \tracingonline\z@%
620   \tracingcommands\z@%
621   \showboxdepth\m@ne%
622   \showboxbreadth\m@ne%
623   \tracingoutput\z@%
624   \errorcontextlines\m@ne%
625   \tracingrestores\z@%
626   \tracingparagraphs\z@%

```

```

627 〈\latexrelease〉 \tracingmacros\z@  

628 〈\latexrelease〉 \tracinglostchars\@ne  

629 〈\latexrelease〉 \tracingpages\z@  

630 〈\latexrelease〉 \tracingstats\z@  

631 〈\latexrelease〉}%
632 〈\latexrelease〉\else  

633 〈\latexrelease〉\def\tracingnone{%
634 〈\latexrelease〉 \tracingassigns\z@  

635 〈\latexrelease〉 \tracingrestores\z@  

636 〈\latexrelease〉 \tracingonline\z@  

637 〈\latexrelease〉 \tracingcommands\z@  

638 〈\latexrelease〉 \showboxdepth\m@ne  

639 〈\latexrelease〉 \showboxbreadth\m@ne  

640 〈\latexrelease〉 \tracingoutput\z@  

641 〈\latexrelease〉 \errorcontextlines\m@ne  

642 〈\latexrelease〉 \tracingnesting\z@  

643 〈\latexrelease〉 \tracingscantokens\z@  

644 〈\latexrelease〉 \tracingifs\z@  

645 〈\latexrelease〉 \tracinggroups\z@  

646 〈\latexrelease〉 \tracingparagraphs\z@  

647 〈\latexrelease〉 \tracingmacros\z@  

648 〈\latexrelease〉 \tracinglostchars\@ne  

649 〈\latexrelease〉 \tracingpages\z@  

650 〈\latexrelease〉 \tracingstats\z@  

651 〈\latexrelease〉}%
652 〈\latexrelease〉\fi  

653 〈\latexrelease〉\EndIncludeInRelease  

654 〈\latexrelease〉  

655 〈\latexrelease〉\IncludeInRelease{0000/00/00}{\tracingnone}%
656 〈\latexrelease〉 {turn off etex tracing}%
657 〈\latexrelease〉\let\tracingnone\@undefined  

658 〈\latexrelease〉\EndIncludeInRelease

```

(End definition for \tracingnone.)

```
\hideoutput
659 〈*2ekernel | \latexrelease〉  

660 〈\latexrelease〉\IncludeInRelease{2015/01/01}{\hideoutput}%
661 〈\latexrelease〉 {hide output from tracing}%
662 \def\hideoutput{%
663   \tracingoutput\z@  

664   \showboxbreadth\m@ne  

665   \showboxdepth\m@ne  

666   \tracingonline\m@ne
667 }%
668 〈\latexrelease〉\EndIncludeInRelease  

669 〈\latexrelease〉  

670 〈\latexrelease〉\IncludeInRelease{0000/00/00}{\hideoutput}%
671 〈\latexrelease〉 {hide output from tracing}%
672 〈\latexrelease〉\let\hideoutput\@undefined  

673 〈\latexrelease〉\EndIncludeInRelease  

674 〈/2ekernel | \latexrelease〉

```

(End definition for \hideoutput.)

L^AT_EX change: \showhyphens Defined later.

Punctuation affects the spacing.

```
675  {*2ekernel}
676  \nonfrenchspacing
677  {/2ekernel}
```

File c

ltvers.dtx

1 Version Identification

First we identify the date and version number of this release of L^AT_EX, and set \everyjob so that it is printed at the start of every L^AT_EX run.

```
\fmtname
\fmtversion
\latexreleaseversion
\patch@level
```

A \patch@level of 0 or higher denotes an official public release. A negative value indicates a candidate release that is not distributed.

If we put code updates into the kernel that are supposed to go into the next release we set the \patch@level to -1 and the \fmtversion / \latexreleaseversion to the dated of the next release (guessed, the real value is not so important and will get corrected when we make the release official).

If the \patch@level is already at -1 we do nothing here and use the \fmtversion date for any new \IncludeInRelease line when we add further code.

Finally, if we do make a public release we either just set the \patch@level to zero (if our initial guess was good) or we also change the date and then have to additionally change to that date on all the \IncludeInRelease statements that used the “guessed” date.

```
1  {*2ekernel}
2  \def\fmtname{LaTeX2e}
3  \edef\fmtversion
4  {/2ekernel}
5  \latexrelease{}\edef\latexreleaseversion
6  {*2ekernel | latexrelease}
7  {2022-06-01}
8  {/2ekernel | latexrelease}
9  {*2ekernel}
10 \def\patch@level{-3}
```

For more fine grain control there is the possibility to name the current development branch. This is only used when the \patch@level is negative (i.e., a pre-release format) and is intended to help us internally when we locally install a format out of some development branch.

```
11 \edef\development@branch@name{develop \the\year-\the\month-\the\day}
```

(End definition for \fmtname and others.)

Check that the format being made is not too old. The error message complains about ‘more than 5 years’ but in fact the error is not triggered until 65 months.

This code is currently not activated as we don’t know if we already got to the last official 2e version (due to staff shortage or due to a successor (think positive:-)).

```
12 \iffalse
13 \def\reserved@a{\#1/#2/#3\@nil}%
14 \count@\year
15 \advance\count@-\#1\relax
16 \multiply\count@ by 12\relax
17 \advance\count@\month
18 \advance\count@-\#2\relax}
19 \expandafter\reserved@a\fmtversion\@nil
```

\count0 is now the age of this file in months. Take a generous definition of ‘year’ so this message is not generated too often.

```

20 \ifnum\count0>65
21   \typeout{^^J%
22 !!! You are attempting to make a LaTeX format from a source file^^J%
23 ! That is more than five years old.^^J%
24 ! ^^J%
25 ! If you enter <return> to scroll past this message then the format^^J%
26 ! will be built, but please consider obtaining newer source files^^J%
27 ! before continuing to build LaTeX.^^J%
28 !!!
29 !
30 }
31 \errhelp{To avoid this error message, obtain new LaTeX sources.}
32 \errmessage{LaTeX source files more than 5 years old!}
33 \fi
34 \let\reserved@a\relax
35 \fi

36 \ifnum0\ifnum\patch@level=0 \ifx\development@branch@name\@empty 1\fi\fi>0 %
37   \everyjob\expandafter{\the\everyjob
38     \typeout{\fmtname\space <\fmtversion>}}
39   \immediate
40   \write16{\fmtname\space<\fmtversion>}
41 \else\ifnum\patch@level>0
42   \everyjob\expandafter{\the\everyjob
43     \typeout{\fmtname\space <\fmtversion> patch level \patch@level}}
44   \immediate
45   \write16{\fmtname\space <\fmtversion> patch level \patch@level}
46 \else
47   \everyjob\expandafter{\the\everyjob
48     \typeout{\fmtname\space <\fmtversion>
49       pre-release-\number-\patch@level\space
50       \ifx\development@branch@name\@undefined \else
51         \ifx\development@branch@name\@empty \else
52           \space (\development@branch@name\space branch)%
53         \fi
54       \fi
55     }}
56   \immediate
57   \write16{\fmtname\space <\fmtversion>
58     pre-release-\number-\patch@level\space
59       \ifx\development@branch@name\@undefined \else
60         \ifx\development@branch@name\@empty \else
61           \space (\development@branch@name\space branch)%
62         \fi
63       \fi
64     }
65   \fi
66 \fi
67 
```

\IncludeInRelease
\EndIncludeInRelease
\@IncludeInRelease
\@IncludeInRelease@
\@gobble@IncludeInRelease
\@check@IncludeInRelease

68 <2ekernel>\let\@currname\@empty

```

69  {*2ekernel | latexrelease}
70  \if@latexrelease\newif\if@includeinrelease
71  \else\fi
72  \def\IncludeInRelease#1{%
73  \if@includeinrelease
74  \PackageError{latexrelease}{mis-matched \IncludeInRelease}%
75  {There is an \string\EndIncludeRelease\space missing}%
76  \else\fi
77  \ifnum0%
78  \ifx\new@moduledate\empty\else 1\fi
79  \ifnum \expandafter\@parse@version#1//00\@nil=0 1\fi
80  =11
81  \expandafter\@firstoftwo
82  \else
83  \expandafter\@secondoftwo
84  \fi
85  {\@finish@module@release{#1}}%
86  {\@kernel@ifnextchar[%]
87  {\@IncludeInRelease{#1}}
88  {\@IncludeInRelease{#1}[#1]}}}
89
90 \def\finish@module@release#1#2#3{%
91  \toks@{[#1] #3}%
92  \begingroup
93  \edef\x{\detokenize\expandafter{\new@modulename}}%
94  \edef\y{\detokenize{#2}}%
95  \expandafter\endgroup
96  \ifx\x\y \else
97  \@latex@error{\noexpand\IncludeInRelease dated #1 in a module is not
98  allowed.\MessageBreak Use a date at least equal to \new@moduledate
99  \space for complete rollback}\@ehd
100 \fi
101 \ifnum\expandafter\@parse@version\new@moduledate//00\@nil
102 >\expandafter\@parse@version\fmtversion//00\@nil
103 \GenericInfo{}{Applying: \the\toks@}%
104 \else
105 \GenericInfo{}{Skipping: \the\toks@}%
106 \expandafter\gobble@finish@module@release
107 \fi}
108 \long\def\gobble@finish@module@release#1\EndModuleRelease{%
109 \EndModuleRelease}

If a specific date has not been specified in \latexrelease use '#1'.

110 \def\@IncludeInRelease#1[#2]{\@IncludeInRelease{#2}}
111 \def\@IncludeInRelease#1#2#3{%
112  \toks@{[#1] #3}%
113  \expandafter\ifx\csname\string#2+\@currname+IIR\endcsname\relax

```

If we roll back and the first patch already match then applying that is actually reapplying what is already in the format, i.e., it is useless and possibly allocating new registers. However, it makes the logic simpler so this is the way it is for now. In theory we could always jump over the first patch because that is only really needed for rolling forward. So maybe one day ...

```

114     \ifnum\expandafter\@parse@version#1//00@nil
115         >\expandafter\@parse@version\fmtversion//00@nil
116         \GenericInfo{}{Skipping: \the\toks@}%
117         \expandafter\expandafter\expandafter\@gobble@IncludeInRelease
118     \else
119         \GenericInfo{}{Applying: \the\toks@}%
120         \@includeinreleasetrue
121         \expandafter\let\csname\string#2+\currname+IIR\endcsname\empty
122         \fi
123     \else
124         \GenericInfo{}{Already applied: \the\toks@}%
125         \expandafter\@gobble@IncludeInRelease
126     \fi
127 }

128 \def\EndIncludeInRelease{%
129 \if@includeinrelease
130   \@includeinreleasefalse
131 \else
132   \PackageError{latexrelease}{mis-matched EndIncludeInRelease}{}%
133 \fi
134 \if@skipping@module
135   \expandafter\new@module@skip
136 \fi}

137 \long\def\@gobble@IncludeInRelease#1\EndIncludeInRelease{%
138   \@includeinreleasefalse
139   \@check@IncludeInRelease#1\IncludeInRelease\@check@IncludeInRelease
140   \@end@check@IncludeInRelease}

141 \long\def\@check@IncludeInRelease#1\IncludeInRelease
142   #2#3\@end@check@IncludeInRelease{%
143   \ifx\@check@IncludeInRelease#2\else
144     \PackageError{latexrelease}{skipped IncludeInRelease for tag \string#2}{}%
145   \fi
146   \if@skipping@module
147     \expandafter\new@module@skip
148   \fi}

```

(End definition for `\IncludeInRelease` and others.)

1.1 Declaring an all-new module

```
\if@skipping@module
\NewModuleRelease
\EndModuleRelease
\new@module@skip
\new@modulename
\new@moduledate
```

When we have a whole new module, we can't roll back to a date where such module exists, otherwise hundreds of "command already defined" errors will pop up. But we can't skip it altogether either, because the module might have changes we still want applied, so a more detailed cherry-picking of code chunks have to be done.

```

149 \let\if@skipping@module\iffalse
150 \def\@skipping@modulerue{\let\if@skipping@module\iftrue}
151 \def\@skipping@modulefalse{\let\if@skipping@module\iffalse}
152 \let\new@modulename\empty
153 \let\new@moduledate\empty
154 \def\NewModuleRelease#1#2#3{%
155   \ifx\new@modulename\empty \else
156     \@latex@error{Nested \noexpand\NewModuleRelease forbidden.}\@ehd \fi

```

```

157  \edef\new@moduledate{\#1}%
158  \edef\new@modulename{\#2}%
159  \GenericInfo{}{BEGIN module: \new@modulename\space (\new@moduledate)}%
160  \GenericInfo{}{ \@spaces\@spaces\@spaces\space#3@gobble}%
161  \ifnum\sourceLaTeXdate<%
162      \expandafter\@parse@version\new@moduledate//00@nil\relax
163  \ifnum\expandafter\@parse@version\fmtversion//00@nil<%
164      \expandafter\@parse@version\new@moduledate//00@nil\relax
165      \GenericInfo{}{Skipping module \new@modulename}%
166      \expandafter\expandafter
167      \expandafter\gobble@finish@module@release
168  \else
169      \GenericInfo{}{Applying module \new@modulename}
170      \@skipping@modulefalse
171      \fi
172  \else
173      \GenericInfo{}{Skipping module \new@modulename}
174      \@skipping@moduletrue
175      \expandafter\new@module@skip
176  \fi}
177 \long\def\new@module@skip#1\IncludeInRelease{%
178     \long\def\reserved@a##1\EndModuleRelease{()}%
179     \if\relax\detokenize\expandafter\{\reserved@a#1{}{}\}\EndModuleRelease\relax
180     \else
181         \@latex@error{Missing mandatory \string\IncludeInRelease{0000/00/00}}\@ehc
182         \expandafter\@secondoftwo
183     \fi
184     \@gobble
185     {\@expandtwoargs\IncludeInRelease
186         {0000/00/00}{\new@modulename}%
187         {ERROR! Emergency recovery}%
188         #1}%
189     \IncludeInRelease}
190 \def\EndModuleRelease{%
191     \ifx\new@modulename\empty
192         \@latex@error{Extra \string\EndModuleRelease.}\@eha
193     \else
194         \GenericInfo{}{END module: \new@modulename\space (\new@moduledate)}%
195         \let\new@modulename\empty
196         \let\new@moduledate\empty
197         \@skipping@modulefalse
198     \fi}

```

(End definition for `\if@skipping@module` and others.)

199 ⟨/2ekernel | latexrelease⟩

File d

ltluatex.dtx

1 Overview

LuaTeX adds a number of engine-specific functions to TeX. Several of these require set up that is best done in the kernel or need related support functions. This file provides *basic* support for LuaTeX at the L^AT_EX 2 _{ε} kernel level plus as a loadable file which can be used with plain TeX and L^AT_EX.

This file contains code for both TeX (to be stored as part of the format) and Lua (to be loaded at the start of each job). In the Lua code, the kernel uses the namespace `luatexbase`.

The following \count registers are used here for register allocation:

```
\e@alloc@attribute@count Attributes (default 258)
\e@alloc@ccodetable@count Category code tables (default 259)
\e@alloc@luafunction@count Lua functions (default 260)
\e@alloc@whatsit@count User whatsits (default 261)
\e@alloc@bytecode@count Lua bytecodes (default 262)
\e@alloc@luachunk@count Lua chunks (default 263)
```

(\count 256 is used for \newmarks allocation and \count 257 is used for \newXeTeXintercharclass with XeTeX, with code defined in `ltfinal.dtx`). With any L^AT_EX 2 _{ε} kernel from 2015 onward these registers are part of the block in the extended area reserved by the kernel (prior to 2015 the L^AT_EX 2 _{ε} kernel did not provide any functionality for the extended allocation area).

2 Core TeX functionality

The commands defined here are defined for possible inclusion in a future L^AT_EX format, however also extracted to the file `ltluatex.tex` which may be used with older L^AT_EX formats, and with plain TeX.

```
\newattribute \newattribute{\langle attribute\rangle}
Defines a named \attribute, indexed from 1 (i.e. \attribute0 is never defined). Attributes initially have the marker value -"7FFFFFFF ('unset') set by the engine.

\newcatcodetable \newcatcodetable{\langle catcodetable\rangle}
Defines a named \catcodetable, indexed from 1 (\catcodetable0 is never assigned). A new catcode table will be populated with exactly those values assigned by IniTeX (as described in the LuaTeX manual).

\newluafunction \newluafunction{\langle function\rangle}
Defines a named \luafunction, indexed from 1. (Lua indexes tables from 1 so \luafunction0 is not available).

\newluacmd \newluacmd{\langle function\rangle}
Like \newluafunction, but defines the command using \luadef instead of just assigning an integer.
```

```

\newprotectedluacmd \newluadef{\function}
    Like \newluacmd, but the defined command is not expandable.
\newwhatsit \newwhatsit{\whatsit}
    Defines a custom \whatsit, indexed from 1.
\newluabytecode \newluabytecode{\bytecode}
    Allocates a number for Lua bytecode register, indexed from 1.
\newluachunkname \newluachunkname{\chunkname}
    Allocates a number for Lua chunk register, indexed from 1. Also enters the name of the
    register (without backslash) into the lua.name table to be used in stack traces.
\catcodetable@initex Predefined category code tables with the obvious assignments. Note that the latex and
\catcodetable@string atletter tables set the full Unicode range to the codes predefined by the kernel.
\catcodetable@latex \setattribute{\attribute}{\value}
\catcodetable@atletter \unsetattribute{\attribute}

\setattribute Set and unset attributes in a manner analogous to \setlength. Note that attributes
\unsetattribute take a marker value when unset so this operation is distinct from setting the value to
zero.

```

3 Plain T_EX interface

The `ltlualatex` interface may be used with plain T_EX using `\input{ltlualatex}`. This inputs `ltlualatex.tex` which inputs `etex.src` (or `etex.sty` if used with L^AT_EX) if it is not already input, and then defines some internal commands to allow the `ltlualatex` interface to be defined.

The `luatexbase` package interface may also be used in plain T_EX, as before, by inputting the package `\input luatexbase.sty`. The new version of `luatexbase` is based on this `ltlualatex` code but implements a compatibility layer providing the interface of the original package.

4 Lua functionality

4.1 Allocators in Lua

```

new_attribute luatexbase.new_attribute(\attribute)
    Returns an allocation number for the \attribute, indexed from 1. The attribute will
    be initialised with the marker value -"7FFFFFFF ('unset'). The attribute allocation se-
    quence is shared with the TEX code but this function does not define a token using
    \attributedef. The attribute name is recorded in the attributes table. A metatable
    is provided so that the table syntax can be used consistently for attributes declared in
    TEX or Lua.

new_whatsit luatexbase.new_whatsit(\whatsit)
    Returns an allocation number for the custom \whatsit, indexed from 1.

new_bytecode luatexbase.new_bytecode(\bytecode)
    Returns an allocation number for a bytecode register, indexed from 1. The optional
    \name argument is just used for logging.

new_chunkname luatexbase.new_chunkname(\chunkname)
    Returns an allocation number for a Lua chunk name for use with \directlua and
    \latelua, indexed from 1. The number is returned and also \name argument is added
    to the lua.name array at that index.

new_luafunction luatexbase.new_luafunction(\functionname)

```

Returns an allocation number for a lua function for use with `\luafunction`, `\lateluafunction`, and `\luadef`, indexed from 1. The optional `<functionname>` argument is just used for logging.

These functions all require access to a named TeX count register to manage their allocations. The standard names are those defined above for access from TeX, e.g. `\e@alloc@attribute@count`, but these can be adjusted by defining the variable `<type>_count_name` before loading `ltluatex.lua`, for example

```
local attribute_count_name = "attributetracker"
require("ltluatex")
```

would use a TeX `\count` (`\countdef`'d token) called `attributetracker` in place of `\e@alloc@attribute@count`.

4.2 Lua access to TeX register numbers

```
registernumber luatexbase.registernumer(<name>)
```

Sometimes (notably in the case of Lua attributes) it is necessary to access a register *by number* that has been allocated by TeX. This package provides a function to look up the relevant number using LuaTeX's internal tables. After for example `\newattribute\myattrib`, `\myattrib` would be defined by (say) `\myattrib=\attribute15`. `luatexbase.registernumer("myattrib")` would then return the register number, 15 in this case. If the string passed as argument does not correspond to a token defined by `\attributedef`, `\countdef` or similar commands, the Lua value `false` is returned.

As an example, consider the input:

```
\newcommand\test[1]{%
\typeout{#1: \expandafter\meaning\csname#1\endcsname^^J
\space\space\space\space
\directlua{tex.write(luatexbase.registernumer("#1") or "bad input")}%
}

\test{undefinedrubbish}

\test{space}

\test{hbox}

\test{@MM}

\test{@tempdima}
\test{@tempdimb}

\test{strutbox}

\test{sixt@@n}

\attributedef\myattr=12
\myattr=200
\test{myattr}
```

If the demonstration code is processed with Lua^LA_TE_X then the following would be produced in the log and terminal output.

```
undefinedrubbish: \relax
    bad input
space: macro:->
    bad input
hbox: \hbox
    bad input
@MM: \mathchar"4E20
    20000
@tempdima: \dimen14
    14
@tempdimb: \dimen15
    15
strutbox: \char"B
    11
sixt@@n: \char"10
    16
myattr: \attribute12
    12
```

Notice how undefined commands, or commands unrelated to registers do not produce an error, just return `false` and so print `bad input` here. Note also that commands defined by `\newbox` work and return the number of the box register even though the actual command holding this number is a `\chardef` defined token (there is no `\boxdef`).

4.3 Module utilities

```
provides_module luatexbase.provides_module(<info>)
```

This function is used by modules to identify themselves; the `info` should be a table containing information about the module. The required field `name` must contain the name of the module. It is recommended to provide a field `date` in the usual L^AT_EX format `yyyy/mm/dd`. Optional fields `version` (a string) and `description` may be used if present. This information will be recorded in the log. Other fields are ignored.

```
module_info luatexbase.module_info(<module>, <text>)
module_warning luatexbase.module_warning(<module>, <text>)
module_error luatexbase.module_error(<module>, <text>)
```

These functions are similar to L^AT_EX's `\PackageError`, `\PackageWarning` and `\PackageInfo` in the way they format the output. No automatic line breaking is done, you may still use `\n` as usual for that, and the name of the package will be prepended to each output line.

Note that `luatexbase.module_error` raises an actual Lua error with `error()`, which currently means a call stack will be dumped. While this may not look pretty, at least it provides useful information for tracking the error down.

4.4 Callback management

```
add_to_callback luatexbase.add_to_callback(<callback>, <function>, <description>) Registers the <function> into the <callback> with a textual <description> of the function. Functions are inserted into the callback in the order loaded.
```

```
remove_from_callback luatexbase.remove_from_callback(<callback>, <description>) Removes the callback
```

function with $\langle description \rangle$ from the $\langle callback \rangle$. The removed function and its description are returned as the results of this function.

`in_callback luatexbase.in_callback($\langle callback \rangle$, $\langle description \rangle$)` Checks if the $\langle description \rangle$ matches one of the functions added to the list for the $\langle callback \rangle$, returning a boolean value.

`disable_callback luatexbase.disable_callback($\langle callback \rangle$)` Sets the $\langle callback \rangle$ to `false` as described in the LuaTeX manual for the underlying `callback.register` built-in. Callbacks will only be set to `false` (and thus be skipped entirely) if there are no functions registered using the callback.

`callback_descriptions` A list of the descriptions of functions registered to the specified callback is returned. `{}` is returned if there are no functions registered.

`create_callback luatexbase.create_callback($\langle name \rangle$, metatype, $\langle default \rangle$)` Defines a user defined callback. The last argument is a default function or `false`.

`call_callback luatexbase.call_callback($\langle name \rangle$, ...)` Calls a user defined callback with the supplied arguments.

5 Implementation

```

1  {*2ekernel | tex | latexrelease}
2  {2ekernel | latexrelease}\ifx\directlua\@undefined\else

```

5.1 Minimum LaTeX version

LaTeX has changed a lot over time. In the kernel support for ancient versions is not provided: trying to build a format with a very old binary therefore gives some information in the log and loading stops. The cut-off selected here relates to the tree-searching behaviour of `require()`: from version 0.60, LaTeX will correctly find Lua files in the `texmf` tree without ‘help’.

```

3  \iflatexrelease\IncludeInRelease{2015/10/01}
4  \iflatexrelease\newluafunction{LuaTeX}%
5  \ifnum\luatexversion<60 %
6  \wlog{*****}
7  \wlog{* LaTeX version too old for ltluatex support *}
8  \wlog{*****}
9  \expandafter\endinput
10 \fi

```

Two simple LaTeX macros from `ltdefns.dtx` have to be defined here because `ltdefns.dtx` is not loaded yet when `ltluatex.dtx` is executed.

```

11 \long\def\@gobble#1{}
12 \long\def\@firstofone#1{#1}

```

5.2 Older LaTeX/Plain TeX setup

```
13 {*}tex
```

Older LaTeX formats don’t have the primitives with ‘native’ names: sort that out. If they already exist this will still be safe.

```

14 \directlua{tex.enableprimitives("",tex.extraprimitives("luatex"))}
15 \ifx\@alloc\@undefined

```

In pre-2014 LaTeX, or plain TeX, load `etex.{sty,src}`.

```

16 \ifx\documentclass\@undefined
17 \ifx\loccount\@undefined

```

```

18      \input{etex.src}%
19      \fi
20      \catcode`@=11 %
21      \outer\expandafter\def\csname newfam\endcsname
22          {\alloc@8\fam\chardef\et@xmaxfam}
23      \else
24          \RequirePackage{etex}
25          \expandafter\def\csname newfam\endcsname
26              {\alloc@8\fam\chardef\et@xmaxfam}
27          \expandafter\let\expandafter\new@mathgroup\csname newfam\endcsname
28      \fi

```

5.2.1 Fixes to etex.src/etex.sty

These could and probably should be made directly in an update to `etex.src` which already has some LuaTeX-specific code, but does not define the correct range for LuaTeX.

2015-07-13 higher range in luatex.

```

29 \edef \et@xmaxregs {\ifx\directlua\undefined 32768\else 65536\fi}
luatex/xetex also allow more math fam.

30 \edef \et@xmaxfam {\ifx\Umathcode\undefined\sixt@@n\else\@cclvi\fi}

31 \count 270=\et@xmaxregs % locally allocates \count registers
32 \count 271=\et@xmaxregs % ditto for \dimen registers
33 \count 272=\et@xmaxregs % ditto for \skip registers
34 \count 273=\et@xmaxregs % ditto for \muskip registers
35 \count 274=\et@xmaxregs % ditto for \box registers
36 \count 275=\et@xmaxregs % ditto for \toks registers
37 \count 276=\et@xmaxregs % ditto for \marks classes

```

and 256 or 16 fam. (Done above due to plain/L^AT_EX differences in l_tluatex.)

```

38 % \outer\def\newfam{\alloc@8\fam\chardef\et@xmaxfam}

```

End of proposed changes to `etex.src`

5.2.2 luatex specific settings

Switch to global cf `luatex.sty` to leave room for inserts not really needed for luatex but possibly most compatible with existing use.

```

39 \expandafter\let\csname newcount\expandafter\expandafter\endcsname
40             \csname globcount\endcsname
41 \expandafter\let\csname newdimen\expandafter\expandafter\endcsname
42             \csname globdimen\endcsname
43 \expandafter\let\csname newskip\expandafter\expandafter\endcsname
44             \csname globskip\endcsname
45 \expandafter\let\csname newbox\expandafter\expandafter\endcsname
46             \csname globbox\endcsname

```

Define `\e@alloc` as in latex (the existing macros in `etex.src` hard to extend to further register types as they assume specific 26x and 27x count range. For compatibility the existing register allocation is not changed.

```

47 \chardef\e@alloc@top=65535
48 \let\e@alloc\chardef\chardef

```

```

49 \def\@alloc#1#2#3#4#5#6{%
50   \global\advance#3\@ne
51   \e@ch@ck{#3}{#4}{#5}#1%
52   \allocationnumber#3\relax
53   \global#2#6\allocationnumber
54   \wlog{\string#6=\string#1\the\allocationnumber}}%
55 \gdef\@ch@ck#1#2#3#4{%
56   \ifnum#1<#2\else
57     \ifnum#1=#2\relax
58       #1\@ccclvi
59       \ifx\count#4\advance#1 10 \fi
60     \fi
61   \ifnum#1<#3\relax
62   \else
63     \errmessage{No room for a new \string#4}%
64   \fi
65 \fi}%

```

Fix up allocations not to clash with `etex.src`.

```

66 \expandafter\csname newcount\endcsname\@alloc@attribute@count
67 \expandafter\csname newcount\endcsname\@alloc@ccodetable@count
68 \expandafter\csname newcount\endcsname\@alloc@luafunction@count
69 \expandafter\csname newcount\endcsname\@alloc@whatsit@count
70 \expandafter\csname newcount\endcsname\@alloc@bytecode@count
71 \expandafter\csname newcount\endcsname\@alloc@luachunk@count

```

End of conditional setup for plain TeX / old L^AT_EX.

```

72 \fi
73 
```

5.3 Attributes

`\newattribute` As is generally the case for the LuaTeX registers we start here from 1. Notably, some code assumes that `\attribute0` is never used so this is important in this case.

```

74 \ifx\@alloc@attribute@count\@undefined
75   \countdef\@alloc@attribute@count=258
76   \@alloc@attribute@count=\z@
77 \fi
78 \def\newattribute#1{%
79   \@alloc@attribute\attributedef
80   \@alloc@attribute@count\m@ne\@alloc@top#1%
81 }

```

(End definition for `\newattribute`.)

`\setattribute` Handy utilities.

```

82 \def\setattribute#1#2{#1=\numexpr#2\relax}
83 \def\unsetattribute#1{#1=-"7FFFFFFF\relax}

```

(End definition for `\setattribute` and `\unsetattribute`.)

5.4 Category code tables

- \newcatcodetable Category code tables are allocated with a limit half of that used by LuaTeX for everything else. At the end of allocation there needs to be an initialization step. Table 0 is already taken (it's the global one for current use) so the allocation starts at 1.

```

84 \ifx\@alloc@ccodetable@count\@undefined
85   \countdef\@alloc@ccodetable@count=259
86   \e@alloc@ccodetable@count=\z@
87 \fi
88 \def\newcatcodetable#1{%
89   \e@alloc@catcodetable\chardef
90     \e@alloc@ccodetable@count\m@ne{"8000}#1%
91   \initcatcodetable\allocationnumber
92 }

```

(End definition for \newcatcodetable.)

- \catcodetable@initex \catcodetable@string \catcodetable@latex \catcodetable@atletter Save a small set of standard tables. The Unicode data is read here in using a parser simplified from that in load-unicode-data: only the nature of letters needs to be detected.

```

93 \newcatcodetable\catcodetable@initex
94 \newcatcodetable\catcodetable@string
95 \begingroup
96 \def\setrangepatcode#1#2#3{%
97   \ifnum#1>#2 %
98     \expandafter\@gobble
99   \else
100     \expandafter\@firstofone
101   \fi
102   {%
103     \catcode#1=#3 %
104     \expandafter\setrangepatcode\expandafter
105       {\number\numexpr#1 + 1\relax}{#2}{#3}
106   }%
107 }
108 \@firstofone{%
109   \catcodetable\catcodetable@initex
110   \catcode0=12 %
111   \catcode13=12 %
112   \catcode37=12 %
113   \setrangepatcode{65}{90}{12}%
114   \setrangepatcode{97}{122}{12}%
115   \catcode92=12 %
116   \catcode127=12 %
117   \savecatcodetable\catcodetable@string
118   \endgroup
119 }%
120 \newcatcodetable\catcodetable@latex
121 \newcatcodetable\catcodetable@atletter
122 \begingroup
123 \def\parseunicodedataI#1;#2;#3;#4\relax{%
124   \parseunicodedataII#1;#3;#2 First>\relax
125 }%
126 \def\parseunicodedataII#1;#2;#3 First>#4\relax{%
127   \ifx\relax#4\relax

```

```

128      \expandafter\parseunicodedataIII
129  \else
130      \expandafter\parseunicodedataIV
131  \fi
132      {#1}#2\relax%
133  }%
134 \def\parseunicodedataIII#1#2#3\relax{%
135  \ifnum 0%
136      \if L#21\fi
137      \if M#21\fi
138      >0 %
139      \catcode"#1=11 %
140  \fi
141 }%
142 \def\parseunicodedataIV#1#2#3\relax{%
143  \read\unicoderead to \unicodedataline
144  \if L#2%
145      \count0="#1 %
146      \expandafter\parseunicodedataV\unicodedataline\relax
147  \fi
148 }%
149 \def\parseunicodedataV#1;#2\relax{%
150  \loop
151      \unless\ifnum\count0>"#1 %
152      \catcode\count0=11 %
153      \advance\count0 by 1 %
154  \repeat
155 }%
156 \def\storedpar{\par}%
157 \chardef\unicoderead=\numexpr\count16 + 1\relax
158 \openin\unicoderead=UnicodeData.txt %
159 \loop\unless\ifeof\unicoderead %
160     \read\unicoderead to \unicodedataline
161     \unless\ifx\unicodedataline\storedpar
162         \expandafter\parseunicodedataI\unicodedataline\relax
163     \fi
164 \repeat
165 \closein\unicoderead
166 \firstofone{%
167     \catcode64=12 %
168     \savecatcodetable\catcodetable@latex
169     \catcode64=11 %
170     \savecatcodetable\catcodetable@atletter
171 }
172 \endgroup

```

(End definition for `\catcodetable@initex` and others.)

5.5 Named Lua functions

`\newluafunction` Much the same story for allocating LuaTeX functions except here they are just numbers so they are allocated in the same way as boxes. Lua indexes from 1 so once again slot 0 is skipped.

173 \ifx\@alloc@luafunction\@undefined

```

174   \countdef\e@alloc@luafunction@count=260
175   \e@alloc@luafunction@count=\z@
176 \fi
177 \def\newluafunction{%
178   \e@alloc@luafunction\e@alloc@chardef
179   \e@alloc@luafunction@count\m@ne\e@alloc@top
180 }

```

(End definition for \newluafunction.)

\newluacmd Additionally two variants are provided to make the passed control sequence call the function directly.

```

181 \def\newluacmd{%
182   \e@alloc@luafunction\luadef
183   \e@alloc@luafunction@count\m@ne\e@alloc@top
184 }
185 \def\newprotectedluacmd{%
186   \e@alloc@luafunction{\protected\luadef}
187   \e@alloc@luafunction@count\m@ne\e@alloc@top
188 }

```

(End definition for \newluacmd and \newprotectedluacmd.)

5.6 Custom whatsits

\newwhatsit These are only settable from Lua but for consistency are definable here.

```

189 \ifx\e@alloc@whatsit@count\@undefined
190   \countdef\e@alloc@whatsit@count=261
191   \e@alloc@whatsit@count=\z@
192 \fi
193 \def\newwhatsit#1{%
194   \e@alloc@whatsit\e@alloc@chardef
195   \e@alloc@whatsit@count\m@ne\e@alloc@top#1%
196 }

```

(End definition for \newwhatsit.)

5.7 Lua bytecode registers

\newluabytecode These are only settable from Lua but for consistency are definable here.

```

197 \ifx\e@alloc@bytecode@count\@undefined
198   \countdef\e@alloc@bytecode@count=262
199   \e@alloc@bytecode@count=\z@
200 \fi
201 \def\newluabytecode#1{%
202   \e@alloc@luabytecode\e@alloc@chardef
203   \e@alloc@bytecode@count\m@ne\e@alloc@top#1%
204 }

```

(End definition for \newluabytecode.)

5.8 Lua chunk registers

\newluachunkname As for bytecode registers, but in addition we need to add a string to the `lua.name` table to use in stack tracing. We use the name of the command passed to the allocator, with no backslash.

```

205 \ifx\@alloc@luachunk@count\@undefined
206   \countdef\@alloc@luachunk@count=263
207   \@alloc@luachunk@count=\z@
208 \fi
209 \def\newluachunkname#1{%
210   \@alloc@luachunk\@alloc@chardef
211   \@alloc@luachunk@count\m@ne\@alloc@top#1%
212   {\escapechar\m@ne
213   \directlua{lua.name[\the\allocationnumber]="\string#1"}%}
214 }

```

(End definition for `\newluachunkname`.)

5.9 Lua loader

Lua code loaded in the format often has to be loaded again at the beginning of every job, so we define a helper which allows us to avoid duplicated code:

```

215 \def\now@and@everyjob#1{%
216   \everyjob\expandafter{\the\everyjob
217   #1%
218 }%
219 #1%
220 }

```

Load the Lua code at the start of every job. For the conversion of T_EX into numbers at the Lua side we need some known registers: for convenience we use a set of systematic names, which means using a group around the Lua loader.

```

221 <2ekernel>\now@and@everyjob{%
222   \begingroup
223   \attributedef\attributezero=0 %
224   \chardef\charzero=0 %

```

Note name change required on older luatex, for hash table access.

```

225   \countdef\CountZero=0 %
226   \dimendef\dimenzero=0 %
227   \mathchardef\mathcharzero=0 %
228   \muskipdef\muskipzero=0 %
229   \skipdef\skipzero=0 %
230   \toksdef\tokszero=0 %
231   \directlua{require("ltluatex")}
232   \endgroup
233 <2ekernel>
234 <latexrelease>\EndIncludeInRelease

235 <latexrelease>\IncludeInRelease{0000/00/00}
236 <latexrelease>{\newluafunction}{LuaTeX}%
237 <latexrelease>\let\@alloc@attribute@count\@undefined
238 <latexrelease>\let\newattribute\@undefined
239 <latexrelease>\let\setattribute\@undefined

```

```

240 〈latexrelease〉\let\unsetattribute\@undefined
241 〈latexrelease〉\let\@alloc@ccodetable@count\@undefined
242 〈latexrelease〉\let\newcatcodetable\@undefined
243 〈latexrelease〉\let\catcodetable@initex\@undefined
244 〈latexrelease〉\let\catcodetable@string\@undefined
245 〈latexrelease〉\let\catcodetable@latex\@undefined
246 〈latexrelease〉\let\catcodetable@atletter\@undefined
247 〈latexrelease〉\let\@alloc@luafunction@count\@undefined
248 〈latexrelease〉\let\newluafunction\@undefined
249 〈latexrelease〉\let\@alloc@luafunction@count\@undefined
250 〈latexrelease〉\let\newwhatsit\@undefined
251 〈latexrelease〉\let\@alloc@whatsit@count\@undefined
252 〈latexrelease〉\let\newluabytocode\@undefined
253 〈latexrelease〉\let\@alloc@bytocode@count\@undefined
254 〈latexrelease〉\let\newluachunkname\@undefined
255 〈latexrelease〉\let\@alloc@luachunk@count\@undefined
256 〈latexrelease〉\directlua{luatexbase.uninstall()}
257 〈latexrelease〉\EndIncludeInRelease

```

In \everyjob, if luaotfloat is available, load it and switch to TU.

```

258 〈latexrelease〉\IncludeInRelease{2017/01/01}%
259 〈latexrelease〉\fontencoding{TU} in everyjob}%
260 〈latexrelease〉\fontencoding{TU}\let\encodingdefault\f@encoding
261 〈latexrelease〉\ifx\directlua\@undefined\else
262 〈2ekernel〉\everyjob\expandafter{%
263 〈2ekernel〉\the\everyjob
264 〈*2ekernel, latexrelease〉
265 \directlua{%
266 if xpcall(function ()%
267 require('luaotfloat-main')%
268 end, texio.write_nl) then %
269 local _void = luaotfloat.main ()%
270 else %
271 texio.write_nl('Error in luaotfloat: reverting to OT1')%
272 tex.print('\string\\def\string\\encodingdefault{OT1}')%
273 end %
274 }%
275 \let\f@encoding\encodingdefault
276 \expandafter\let\csname ver@luaotfloat.sty\endcsname\fmtversion
277 〈/2ekernel, latexrelease〉
278 〈latexrelease〉\fi
279 〈2ekernel〉}
280 〈latexrelease〉\EndIncludeInRelease
281 〈latexrelease〉\IncludeInRelease{0000/00/00}%
282 〈latexrelease〉\fontencoding{TU} in everyjob}%
283 〈latexrelease〉\fontencoding{OT1}\let\encodingdefault\f@encoding
284 〈latexrelease〉\EndIncludeInRelease
285 〈2ekernel | latexrelease〉\fi
286 〈/2ekernel | tex | latexrelease〉

```

5.10 Lua module preliminaries

```
287 〈*lua〉
```

Some set up for the Lua module which is needed for all of the Lua functionality added here.

luatexbase Set up the table for the returned functions. This is used to expose all of the public functions.

```
288 luatexbase      = luatexbase or { }
289 local luatexbase = luatexbase
```

(*End definition for luatexbase.*)

Some Lua best practice: use local versions of functions where possible.

```
290 local string_gsub      = string.gsub
291 local tex_count         = tex.count
292 local tex_setattribute = tex.setattribute
293 local tex_setcount      = tex.setcount
294 local texio_write_nl   = texio.write_nl
295 local flush_list        = node.flush_list

296 local luatexbase_warning
297 local luatexbase_error
```

5.11 Lua module utilities

5.11.1 Module tracking

modules To allow tracking of module usage, a structure is provided to store information and to return it.

```
298 local modules = modules or { }
```

(*End definition for modules.*)

provides_module Local function to write to the log.

```
299 local function luatexbase_log(text)
300   texio_write_nl("log", text)
301 end
```

Modelled on \ProvidesPackage, we store much the same information but with a little more structure.

```
302 local function provides_module(info)
303   if not (info and info.name) then
304     luatexbase_error("Missing module name for provides_module")
305   end
306   local function spaced(text)
307     return text and (" " .. text) or ""
308   end
309   luatexbase_log(
310     "Lua module: " .. info.name
311     .. spaced(info.date)
312     .. spaced(info.version)
313     .. spaced(info.description)
314   )
315   modules[info.name] = info
316 end
317 luatexbase.provides_module = provides_module
```

(*End definition for provides_module.*)

5.11.2 Module messages

There are various warnings and errors that need to be given. For warnings we can get exactly the same formatting as from TeX. For errors we have to make some changes. Here we give the text of the error in the L^AT_EX format then force an error from Lua to halt the run. Splitting the message text is done using \n which takes the place of \MessageBreak.

First an auxiliary for the formatting: this measures up the message leader so we always get the correct indent.

```

318 local function msg_format(mod, msg_type, text)
319   local leader = ""
320   local cont
321   local first_head
322   if mod == "LaTeX" then
323     cont = string.gsub(leader, ".", " ")
324     first_head = leader .. "LaTeX: "
325   else
326     first_head = leader .. "Module " .. msg_type
327     cont = "(" .. mod .. ")"
328     .. string.gsub(first_head, ".", " ")
329     first_head = leader .. "Module " .. mod .. " " .. msg_type .. ":" ..
330   end
331   if msg_type == "Error" then
332     first_head = "\n" .. first_head
333   end
334   if string.sub(text,-1) ~= "\n" then
335     text = text .. " "
336   end
337   return first_head .. " "
338   .. string.gsub(
339     text
340   .. "on input line "
341     .. tex.inputlineno, "\n", "\n" .. cont .. " "
342   )
343   .. "\n"
344 end

```

`module_info` Write messages.

```

module_warning 345 local function module_info(mod, text)
module_error    346   texio_write_nl("log", msg_format(mod, "Info", text))
347 end
348 luatexbase.module_info = module_info
349 local function module_warning(mod, text)
350   texio_write_nl("term and log", msg_format(mod, "Warning", text))
351 end
352 luatexbase.module_warning = module_warning
353 local function module_error(mod, text)
354   error(msg_format(mod, "Error", text))
355 end
356 luatexbase.module_error = module_error

```

(End definition for `module_info`, `module_warning`, and `module_error`.)

Dedicated versions for the rest of the code here.

```

357 function luatexbase_warning(text)
358     module_warning("luatexbase", text)
359 end
360 function luatexbase_error(text)
361     module_error("luatexbase", text)
362 end

```

5.12 Accessing register numbers from Lua

Collect up the data from the \TeX level into a Lua table: from version 0.80, $\text{Lua}\text{\TeX}$ makes that easy.

```

363 local luaregisterbasetable = { }
364 local registermap = {
365     attributezero = "assign_attr"      ,
366     charzero     = "char_given"       ,
367     CountZero    = "assign_int"       ,
368     dimenzero    = "assign_dimen"     ,
369     mathcharzero = "math_given"       ,
370     muskipzero   = "assign_mu_skip"  ,
371     skipzero     = "assign_skip"     ,
372     tokszero    = "assign_toks"      ,
373 }
374 local createtoken
375 if tex.luatexversion > 81 then
376     createtoken = token.create
377 elseif tex.luatexversion > 79 then
378     createtoken = newtoken.create
379 end
380 local hashtokens = tex.hashtokens()
381 local luatexversion = tex.luatexversion
382 for i,j in pairs (registermap) do
383     if luatexversion < 80 then
384         luaregisterbasetable[hashtokens[i][1]] =
385             hashtokens[i][2]
386     else
387         luaregisterbasetable[j] = createtoken(i).mode
388     end
389 end

```

registernumber Working out the correct return value can be done in two ways. For older $\text{Lua}\text{\TeX}$ releases it has to be extracted from the `hashtokens`. On the other hand, newer $\text{Lua}\text{\TeX}$'s have `newtoken`, and whilst `.mode` isn't currently documented, Hans Hagen pointed to this approach so we should be OK.

```

390 local registernumber
391 if luatexversion < 80 then
392     function registernumber(name)
393         local nt = hashtokens[name]
394         if(nt and luaregisterbasetable[nt[1]]) then
395             return nt[2] - luaregisterbasetable[nt[1]]
396         else
397             return false
398         end
399     end

```

```

400 else
401     function registernumber(name)
402         local nt = createtoken(name)
403         if(luaregisterbasetable[nt.cmdname]) then
404             return nt.mode - luaregisterbasetable[nt.cmdname]
405         else
406             return false
407         end
408     end
409 end
410 luatexbase.registernumber = registernumber

```

(End definition for registernumber.)

5.13 Attribute allocation

`new_attribute` As attributes are used for Lua manipulations its useful to be able to assign from this end.

```

411 local attributes=setmetatable(
412 {}, {
413     __index = function(t,key)
414         return registernumber(key) or nil
415     end}
416 )
417 )
418 luatexbase.attributes = attributes

419 local attribute_count_name =
420         attribute_count_name or "e@alloc@attribute@count"
421 local function new_attribute(name)
422     tex_setcount("global", attribute_count_name,
423                 tex_count[attribute_count_name] + 1)
424     if tex_count[attribute_count_name] > 65534 then
425         luatexbase_error("No room for a new \\attribute")
426     end
427     attributes[name]= tex_count[attribute_count_name]
428     luatexbase_log("Lua-only attribute " .. name .. " = " ..
429                     tex_count[attribute_count_name])
430     return tex_count[attribute_count_name]
431 end
432 luatexbase.new_attribute = new_attribute

```

(End definition for new_attribute.)

5.14 Custom whatsit allocation

`new_whatsit` Much the same as for attribute allocation in Lua.

```

433 local whatsit_count_name = whatsit_count_name or "e@alloc@whatsit@count"
434 local function new_whatsit(name)
435     tex_setcount("global", whatsit_count_name,
436                 tex_count[whatsit_count_name] + 1)
437     if tex_count[whatsit_count_name] > 65534 then
438         luatexbase_error("No room for a new custom whatsit")
439     end
440     luatexbase_log("Custom whatsit " .. (name or "") .. " = " ..

```

```

441             tex_count[whatsit_count_name])
442     return tex_count[whatsit_count_name]
443   end
444 luatexbase.new_whatsit = new_whatsit

```

(End definition for `new_whatsit`.)

5.15 Bytecode register allocation

`new_bytecode` Much the same as for attribute allocation in Lua. The optional $\langle name \rangle$ argument is used in the log if given.

```

445 local bytecode_count_name =
446           bytecode_count_name or "e@alloc@bytecode@count"
447 local function new_bytecode(name)
448   tex_setcount("global", bytecode_count_name,
449               tex_count[bytecode_count_name] + 1)
450   if tex_count[bytecode_count_name] > 65534 then
451     luatexbase_error("No room for a new bytecode register")
452   end
453   luatexbase_log("Lua bytecode " .. (name or "") .. " = " ..
454                 tex_count[bytecode_count_name])
455   return tex_count[bytecode_count_name]
456 end
457 luatexbase.new_bytecode = new_bytecode

```

(End definition for `new_bytecode`.)

5.16 Lua chunk name allocation

`new_chunkname` As for bytecode registers but also store the name in the `lua.name` table.

```

458 local chunkname_count_name =
459           chunkname_count_name or "e@alloc@luachunk@count"
460 local function new_chunkname(name)
461   tex_setcount("global", chunkname_count_name,
462               tex_count[chunkname_count_name] + 1)
463   local chunkname_count = tex_count[chunkname_count_name]
464   chunkname_count = chunkname_count + 1
465   if chunkname_count > 65534 then
466     luatexbase_error("No room for a new chunkname")
467   end
468   lua.name[chunkname_count]=name
469   luatexbase_log("Lua chunkname " .. (name or "") .. " = " ..
470                 chunkname_count .. "\n")
471   return chunkname_count
472 end
473 luatexbase.new_chunkname = new_chunkname

```

(End definition for `new_chunkname`.)

5.17 Lua function allocation

`new_luafunction` Much the same as for attribute allocation in Lua. The optional `<name>` argument is used in the log if given.

```
474 local luafunction_count_name =
475           luafunction_count_name or "e@alloc@luafunction@count"
476 local function new_luafunction(name)
477   tex_setcount("global", luafunction_count_name,
478               tex_count[luafunction_count_name] + 1)
479   if tex_count[luafunction_count_name] > 65534 then
480     luatexbase_error("No room for a new luafunction register")
481   end
482   luatexbase_log("Lua function " .. (name or "") .. " = " ..
483                 tex_count[luafunction_count_name])
484   return tex_count[luafunction_count_name]
485 end
486 luatexbase.new_luafunction = new_luafunction
```

(End definition for `new_luafunction`.)

5.18 Lua callback management

The native mechanism for callbacks in LuaTeX allows only one per function. That is extremely restrictive and so a mechanism is needed to add and remove callbacks from the appropriate hooks.

5.18.1 Housekeeping

The main table: keys are callback names, and values are the associated lists of functions. More precisely, the entries in the list are tables holding the actual function as `func` and the identifying description as `description`. Only callbacks with a non-empty list of functions have an entry in this list.

```
487 local callbacklist = callbacklist or { }
```

Numerical codes for callback types, and name-to-value association (the table keys are strings, the values are numbers).

```
488 local list, data, exclusive, simple, reverselist = 1, 2, 3, 4, 5
489 local types    = {
490   list      = list,
491   data      = data,
492   exclusive = exclusive,
493   simple    = simple,
494   reverselist = reverselist,
495 }
```

Now, list all predefined callbacks with their current type, based on the LuaTeX manual version 1.01. A full list of the currently-available callbacks can be obtained using

```
\directlua{
  for i,_ in pairs(callback.list()) do
    texio.write_nl("- " .. i)
  end
}
\bye
```

in plain LuaTeX. (Some undocumented callbacks are omitted as they are to be removed.)

```
496 local callbacktypes = callbacktypes or {
```

Section 8.2: file discovery callbacks.

```
497   find_read_file      = exclusive,
498   find_write_file     = exclusive,
499   find_font_file      = data,
500   find_output_file    = data,
501   find_format_file    = data,
502   find_vf_file        = data,
503   find_map_file       = data,
504   find_enc_file       = data,
505   find_pk_file        = data,
506   find_data_file      = data,
507   find_opentype_file  = data,
508   find_truetype_file  = data,
509   find_type1_file     = data,
510   find_image_file     = data,
511
511   open_read_file      = exclusive,
512   read_font_file      = exclusive,
513   read_vf_file        = exclusive,
514   read_map_file       = exclusive,
515   read_enc_file       = exclusive,
516   read_pk_file        = exclusive,
517   read_data_file      = exclusive,
518   read_truetype_file  = exclusive,
519   read_type1_file     = exclusive,
520   read_opentype_file  = exclusive,
```

Not currently used by luatex but included for completeness. may be used by a font handler.

```
521   find_cidmap_file   = data,
522   read_cidmap_file   = exclusive,
```

Section 8.3: data processing callbacks.

```
523   process_input_buffer = data,
524   process_output_buffer = data,
525   process_jobname      = data,
```

Section 8.4: node list processing callbacks.

```
526   contribute_filter   = simple,
527   buildpage_filter    = simple,
528   build_page_insert   = exclusive,
529   pre_linebreak_filter = list,
530   linebreak_filter    = exclusive,
531   append_to_vlist_filter = exclusive,
532   post_linebreak_filter = reverselist,
533   hpack_filter        = list,
534   vpack_filter        = list,
535   hpack_quality       = exclusive,
536   vpack_quality       = exclusive,
537   pre_output_filter   = list,
538   process_rule         = exclusive,
539   hyphenate           = simple,
```

```

540 ligaturing           = simple,
541 kerning              = simple,
542 insert_local_par     = simple,
543 pre_mlist_to_hlist_filter = list,
544 mlist_to_hlist        = exclusive,
545 post_mlist_to_hlist_filter = reverselist,
546 new_graf              = exclusive,

```

Section 8.5: information reporting callbacks.

```

547 pre_dump             = simple,
548 start_run             = simple,
549 stop_run              = simple,
550 start_page_number    = simple,
551 stop_page_number     = simple,
552 show_error_hook       = simple,
553 show_warning_message = simple,
554 show_error_message   = simple,
555 show_lua_error_hook  = simple,
556 start_file            = simple,
557 stop_file             = simple,
558 call_edit              = simple,
559 finish_synctex       = simple,
560 wrapup_run            = simple,

```

Section 8.6: PDF-related callbacks.

```

561 finish_pdffile       = data,
562 finish_pdfpage       = data,
563 page_objnum_provider = data,
564 page_order_index     = data,
565 process_pdf_image_content = data,

```

Section 8.7: font-related callbacks.

```

566 define_font           = exclusive,
567 glyph_info             = exclusive,
568 glyph_not_found       = exclusive,
569 glyph_stream_provider = exclusive,
570 make_extensible        = exclusive,
571 font_descriptor_objnum_provider = exclusive,
572 input_level_string     = exclusive,
573 provide_charproc_data = exclusive,
574 }
575 luatexbase.callbacktypes=callbacktypes

```

`callback.register` Save the original function for registering callbacks and prevent the original being used. The original is saved in a place that remains available so other more sophisticated code can override the approach taken by the kernel if desired.

```

576 local callback_register = callback_register or callback.register
577 function callback.register()
578   luatexbase_error("Attempt to use callback.register() directly\n")
579 end

```

(End definition for `callback.register`.)

5.18.2 Handlers

The handler function is registered into the callback when the first function is added to this callback's list. Then, when the callback is called, the handler takes care of running all functions in the list. When the last function is removed from the callback's list, the handler is unregistered.

More precisely, the functions below are used to generate a specialized function (closure) for a given callback, which is the actual handler.

The way the functions are combined together depends on the type of the callback. There are currently 4 types of callback, depending on the calling convention of the functions the callback can hold:

simple is for functions that don't return anything: they are called in order, all with the same argument;

data is for functions receiving a piece of data of any type except node list head (and possibly other arguments) and returning it (possibly modified): the functions are called in order, and each is passed the return value of the previous (and the other arguments untouched, if any). The return value is that of the last function;

list is a specialized variant of *data* for functions filtering node lists. Such functions may return either the head of a modified node list, or the boolean values **true** or **false**. The functions are chained the same way as for *data* except that for the following. If one function returns **false**, then **false** is immediately returned and the following functions are *not* called. If one function returns **true**, then the same head is passed to the next function. If all functions return **true**, then **true** is returned, otherwise the return value of the last function not returning **true** is used.

reverselist is a specialized variant of *list* which executes functions in inverse order.

exclusive is for functions with more complex signatures; functions in this type of callback are *not* combined: An error is raised if a second callback is registered.

Handler for **data** callbacks.

```
580 local function data_handler(name)
581   return function(data, ...)
582     for _,i in ipairs(callbacklist[name]) do
583       data = i.func(data,...)
584     end
585     return data
586   end
587 end
```

Default for user-defined **data** callbacks without explicit default.

```
588 local function data_handler_default(value)
589   return value
590 end
```

Handler for **exclusive** callbacks. We can assume `callbacklist[name]` is not empty: otherwise, the function wouldn't be registered in the callback any more.

```
591 local function exclusive_handler(name)
592   return function(...)
593     return callbacklist[name][1].func(...)
594   end
595 end
```

Handler for list callbacks.

```
596 local function list_handler(name)
597   return function(head, ...)
598     local ret
599     for _,i in ipairs(callbacklist[name]) do
600       ret = i.func(head, ...)
601       if ret == false then
602         luatexbase_warning(
603           "Function '" .. i.description .. "' returned false\n"
604           .. "in callback '" .. name .. "'"
605         )
606         return false
607       end
608       if ret ~= true then
609         head = ret
610       end
611     end
612     return head
613   end
614 end
```

Default for user-defined list and reverselist callbacks without explicit default.

```
615 local function list_handler_default(head)
616   return head
617 end
```

Handler for reverselist callbacks.

```
618 local function reverselist_handler(name)
619   return function(head, ...)
620     local ret
621     local callbacks = callbacklist[name]
622     for i = #callbacks, 1, -1 do
623       local cb = callbacks[i]
624       ret = cb.func(head, ...)
625       if ret == false then
626         luatexbase_warning(
627           "Function '" .. cb.description .. "' returned false\n"
628           .. "in callback '" .. name .. "'"
629         )
630         return false
631       end
632       if ret ~= true then
633         head = ret
634       end
635     end
636     return head
637   end
638 end
```

Handler for simple callbacks.

```
639 local function simple_handler(name)
640   return function(...)
641     for _,i in ipairs(callbacklist[name]) do
642       i.func(...)
643     end
644   end
645 end
```

```

644     end
645 end
Default for user-defined simple callbacks without explicit default.
646 local function simple_handler_default()
647 end
Keep a handlers table for indexed access and a table with the corresponding default
functions.
648 local handlers = {
649     [data]      = data_handler,
650     [exclusive] = exclusive_handler,
651     [list]       = list_handler,
652     [reverselist] = reverselist_handler,
653     [simple]     = simple_handler,
654 }
655 local defaults = {
656     [data]      = data_handler_default,
657     [exclusive] = nil,
658     [list]       = list_handler_default,
659     [reverselist] = list_handler_default,
660     [simple]     = simple_handler_default,
661 }

```

5.18.3 Public functions for callback management

Defining user callbacks perhaps should be in package code, but impacts on `add_to_callback`. If a default function is not required, it may be declared as `false`. First we need a list of user callbacks.

```

662 local user_callbacks_defaults = {
663     pre_mlist_to_hlist_filter = list_handler_default,
664     mlist_to_hlist = node.mlist_to_hlist,
665     post_mlist_to_hlist_filter = list_handler_default,
666 }

```

`create_callback` The allocator itself.

```

667 local function create_callback(name, ctype, default)
668     local ctype_id = types[ctype]
669     if not name or name == ""
670     or not ctype_id
671     then
672         luatexbase_error("Unable to create callback:\n" ..
673                         "valid callback name and type required")
674     end
675     if callbacktypes[name] then
676         luatexbase_error("Unable to create callback '" .. name ..
677                         "' :\ncallback is already defined")
678     end
679     default = default or defaults[ctype_id]
680     if not default then
681         luatexbase_error("Unable to create callback '" .. name ..
682                         "' :\ndefault is required for '" .. ctype ..
683                         "' callbacks")
684     elseif type (default) ~= "function" then

```

```

685     luatexbase_error("Unable to create callback '" .. name ..
686                             "':\\ndefault is not a function")
687   end
688   user_callbacks_defaults[name] = default
689   callbacktypes[name] = ctype_id
690 end
691 luatexbase.create_callback = create_callback
(End definition for create_callback.)
```

`call_callback` Call a user defined callback. First check arguments.

```

692 local function call_callback(name,...)
693   if not name or name == "" then
694     luatexbase_error("Unable to create callback:\\n" ..
695                           "valid callback name required")
696   end
697   if user_callbacks_defaults[name] == nil then
698     luatexbase_error("Unable to call callback '" .. name ..
699                           .. "'\\nunknown or empty")
700   end
701   local l = callbacklist[name]
702   local f
703   if not l then
704     f = user_callbacks_defaults[name]
705   else
706     f = handlers[callbacktypes[name]](name)
707   end
708   return f(...)
709 end
710 luatexbase.call_callback=call_callback
(End definition for call_callback.)
```

`add_to_callback` Add a function to a callback. First check arguments.

```

711 local function add_to_callback(name, func, description)
712   if not name or name == "" then
713     luatexbase_error("Unable to register callback:\\n" ..
714                           "valid callback name required")
715   end
716   if not callbacktypes[name] or
717     type(func) ~= "function" or
718     not description or
719     description == "" then
720     luatexbase_error(
721       "Unable to register callback.\\n\\n"
722       .. "Correct usage:\\n"
723       .. "add_to_callback(<callback>, <function>, <description>)"
724     )
725   end
```

Then test if this callback is already in use. If not, initialise its list and register the proper handler.

```

726   local l = callbacklist[name]
727   if l == nil then
728     l = { }
729     callbacklist[name] = l
```

If it is not a user defined callback use the primitive callback register.

```
730     if user_callbacks_defaults[name] == nil then
731         callback_register(name, handlers[callbacktypes[name]](name))
732     end
733 end
```

Actually register the function and give an error if more than one `exclusive` one is registered.

```
734     local f = {
735         func      = func,
736         description = description,
737     }
738     local priority = #l + 1
739     if callbacktypes[name] == exclusive then
740         if #l == 1 then
741             luatexbase_error(
742                 "Cannot add second callback to exclusive function\n" ..
743                 name .. "'")
744         end
745     end
746     table.insert(l, priority, f)
```

Keep user informed.

```
747     luatexbase_log(
748         "Inserting '" .. description .. "' at position "
749         .. priority .. " in '" .. name .. "'."
750     )
751 end
752 luatexbase.add_to_callback = add_to_callback
```

(End definition for `add_to_callback`.)

`remove_from_callback` Remove a function from a callback. First check arguments.

```
753 local function remove_from_callback(name, description)
754     if not name or name == "" then
755         luatexbase_error("Unable to remove function from callback:\n" ..
756                         "valid callback name required")
757     end
758     if not callbacktypes[name] or
759         not description or
760         description == "" then
761         luatexbase_error(
762             "Unable to remove function from callback.\n\n"
763             .. "Correct usage:\n"
764             .. "remove_from_callback(<callback>, <description>)"
765         )
766     end
767     local l = callbacklist[name]
768     if not l then
769         luatexbase_error(
770             "No callback list for '" .. name .. "'\n")
771     end
```

Loop over the callback's function list until we find a matching entry. Remove it and check if the list is empty: if so, unregister the callback handler.

```

772 local index = false
773 for i,j in ipairs(l) do
774   if j.description == description then
775     index = i
776     break
777   end
778 end
779 if not index then
780   luatexbase_error(
781     "No callback \" .. description .. \" registered for \" ..
782     name .. \"\n")
783 end
784 local cb = l[index]
785 table.remove(l, index)
786 luatexbase_log(
787   "Removing \" .. description .. \" from \" .. name .. \"."
788 )
789 if #l == 0 then
790   callbacklist[name] = nil
791   if user_callbacks_defaults[name] == nil then
792     callback_register(name, nil)
793   end
794 end
795 return cb.func,cb.description
796 end
797 luatexbase.remove_from_callback = remove_from_callback

```

(End definition for remove_from_callback.)

in_callback Look for a function description in a callback.

```

798 local function in_callback(name, description)
799   if not name
800     or name == ""
801     or not callbacklist[name]
802     or not callbacktypes[name]
803     or not description then
804       return false
805     end
806   for _, i in pairs(callbacklist[name]) do
807     if i.description == description then
808       return true
809     end
810   end
811   return false
812 end
813 luatexbase.in_callback = in_callback

```

(End definition for in_callback.)

disable_callback As we subvert the engine interface we need to provide a way to access this functionality.

```

814 local function disable_callback(name)
815   if(callbacklist[name] == nil) then
816     callback_register(name, false)
817   else
818     luatexbase_error("Callback list for \" .. name .. \" not empty")

```

```

819     end
820   end
821   luatexbase.disable_callback = disable_callback

```

(End definition for `disable_callback`.)

`callback_descriptions` List the descriptions of functions registered for the given callback.

```

822 local function callback_descriptions (name)
823   local d = {}
824   if not name
825     or name == ""
826     or not callbacklist[name]
827     or not callbacktypes[name]
828   then
829     return d
830   else
831     for k, i in pairs(callbacklist[name]) do
832       d[k]= i.description
833     end
834   end
835   return d
836 end
837 luatexbase.callback_descriptions =callback_descriptions

```

(End definition for `callback_descriptions`.)

`uninstall` Unlike at the TeX level, we have to provide a back-out mechanism here at the same time as the rest of the code. This is not meant for use by anything other than `latexrelease`: as such this is *deliberately* not documented for users!

```

838 local function uninstall()
839   module_info(
840     "luatexbase",
841     "Uninstalling kernel luatexbase code"
842   )
843   callback.register = callback_register
844   luatexbase = nil
845 end
846 luatexbase.uninstall = uninstall

```

(End definition for `uninstall`.)

`mlist_to_hlist` To emulate these callbacks, the “real” `mlist_to_hlist` is replaced by a wrapper calling the wrappers before and after.

```

847 callback_register("mlist_to_hlist", function(head, display_type, need_penalties)
848   local current = call_callback("pre_mlist_to_hlist_filter", head, display_type, need_penalties)
849   if current == false then
850     flush_list(head)
851     return nil
852   end
853   current = call_callback("mlist_to_hlist", current, display_type, need_penalties)
854   local post = call_callback("post_mlist_to_hlist_filter", current, display_type, need_penalties)
855   if post == false then
856     flush_list(current)
857     return nil

```

```
858     end
859     return post
860 end)

(End definition for mlist_to_hlist.)
```

861 ⟨/lua⟩

Reset the catcode of @.

```
862 ⟨tex⟩\catcode`@=\et atcatcode\relax
```

File e

ltxexpl.dtx

1 expl3-dependent code

1.1 Loader

\@kernel@after@enddocument
\@kernel@after@enddocument@afterlastpage

These two kernel hooks are used by the shipout code. They are defined earlier here because the lhooks code adds material to them.

```
1  {*2ekernel | latexrelease}
2  <{latexrelease}>\IncludeInRelease{2020/10/01}%
3  <{latexrelease}> {kernel@enddocument hooks}{Define several kernel hooks}
```

We only initialize these kernel hooks if they are not already existing. Otherwise they would be set to \empty on rollback which would be wrong because code that has been added to them may still have to be executed in the rollback situation. Instead code that writes to them needs to handle the rollback as needed. It is likely that we have to change that approach in the future, but for now it should do. (It is enough to test only for the existence of one hook, as all got added at the same time.)

```
4  \ifx\@kernel@after@enddocument\@undefined
5    \let\@kernel@after@enddocument\empty
6    \let\@kernel@after@enddocument@afterlastpage\empty
```

For the similar reasons we also define those that are used in \document because they too get material added to in early modules.

```
7  \let\@kernel@before@begindocument\empty
8  \let\@kernel@after@begindocument\empty
9  \fi
10 <{latexrelease}>\EndIncludeInRelease
11 <{latexrelease}>\IncludeInRelease{0000/00/00}%
12 <{latexrelease}> {kernel@enddocument hooks}{Define several kernel hooks}
13 <{latexrelease}>\let\@kernel@after@enddocument\@undefined
14 <{latexrelease}>\let\@kernel@after@enddocument@afterlastpage\@undefined
15 <{latexrelease}>\let\@kernel@before@begindocument\@undefined
16 <{latexrelease}>\let\@kernel@after@begindocument\@undefined
17 <{/2ekernel | latexrelease}>
18 <{latexrelease}>\EndIncludeInRelease
```

(End definition for \@kernel@after@enddocument and others.)

First define some blank commands, so that in case something goes wrong while loading expl3, we won't get strange Undefined control sequence errors.

```
19  {*2ekernel | latexrelease}
20  <{latexrelease}>\IncludeInRelease{2020/10/01}%
21  <{latexrelease}> {\@expl@sys@load@backend@@}{Roll forward support}%
22  \def\reserved@a{\ifdefined#1\else\def#1{}\fi}
23  \reserved@a\@expl@sys@load@backend@@
24  \reserved@a\@expl@push@filename@@
25  \reserved@a\@expl@push@filename@aux@@
26  \reserved@a\@expl@pop@filename@@
27  <{latexrelease}>\EndIncludeInRelease
28 <{/2ekernel | latexrelease}>
```

Create a hook for last-minute expl3 material.

```
29  {*2ekernel}
30  \def\@expl@finalise@setup@@{}
31  (/2ekernel}
```

Now define some basics to support loading expl3. These macros can be defined here safely, because they are redefined later on by the kernel, so we define simpler versions just to suit our needs.

```
32  {*2ekernel}
33  \long\def\@gobble#1{}
34  \long\def\@firstofone#1{#1}
35  \long\def\@firstoftwo#1#2{#1}
36  \long\def\@secondoftwo#1#2{#2}
37  \long\def\IfFileExists#1{%
38    \openin\@inputcheck"#1" %
39    \ifeof\@inputcheck
40      \expandafter\@secondoftwo
41    \else
42      \closein\@inputcheck
43      \expandafter\@firstoftwo
44    \fi}
45  \long\def\@ifnextchar#1#2#3{%
46    \let\reserved@d=#1%
47    \def\reserved@a{#2}%
48    \def\reserved@b{#3}%
49    \futurelet\@let@token\@ifnch}
50  \def\@ifnch{%
51    \ifx\@let@token\reserved@d
52      \expandafter\reserved@a
53    \else
54      \expandafter\reserved@b
55    \fi}
56  (/2ekernel}
```

If we are doing a rollback with a format containing expl3 we aren't reloading it as that creates havoc. This may need a refined version!

```
57  {*2ekernel | latexrelease}
58  <| latexrelease> \IncludeInRelease{2020/10/01}%
59  <| latexrelease>           {expl3}{Pre-load expl3}%
60  \expandafter\ifx\csname tex\string _let:D\endcsname\relax
61    \expandafter\@firstofone
62  \else
63    \GenericInfo{}{Skipping: expl3 code already part of the format}%
64  <2ekernel>  \expandafter\endinput
65  <| latexrelease>  \expandafter\@gobble
66  \fi
```

Check for the required primitive/engine support and the existence of a loader.

```
67  {%
68    \IfFileExists{expl3.ltx}%
69    {%
70      \ifnum0%
71        \ifdefined\pdffilesize 1\fi
72        \ifdefined\filesize 1\fi
73        \ifdefined\luatexversion\ifnum\luatexversion>94 1\fi\fi
74    }%
```

```

74          \ifdefined\kanjiskip 1\fi
75          >0 %
76          \expandafter\@firstofone
77          \else

```

In 2ekernel mode, an error is fatal and building the format is aborted. Use `\batchmode \read -1` to `\tokenlist`, which errors with

```
! Emergency stop. (cannot \read from terminal in nonstop modes)
```

and aborts the TeX run. In latexrelease mode, raise an error and do nothing. Both ways, the error message shows the minimum expl3 engine requirements.

```

78 <2ekernel>      \def~{ }\def\MessageBreak{^^J~~~~~}%
79 <2ekernel>      \errmessage{LaTeX Error:
80 <latexrelease>    \@latex@error{%
81           LaTeX requires the e-TeX primitives and additional\MessageBreak
82           functionality available in the engines:\MessageBreak
83           - pdfTeX v1.40\MessageBreak
84           - XeTeX v0.99992\MessageBreak
85           - LuaTeX v0.95\MessageBreak
86           - e-(u)pTeX mid-2012\MessageBreak
87           or later%
88 <|latexrelease>   } \@ehd \expandafter\@gobble
89 <2ekernel>      } \batchmode \read -1 to \reserved@a
90           \fi
91       }
92       {%
93 <*2ekernel>
94       \errmessage{LaTeX requires expl3}%
95       \batchmode \read -1 to \reserved@a
96 </2ekernel>

```

We do not support a roll forward across 2019. You need to start with 2019 if you want to get to 2020 or beyond.

```

97 <*latexrelease>
98     \@latex@warning@no@line
99     {You need a format that already contains a recent\MessageBreak
100      expl3 as part of the kernel, e.g. at least a kernel\MessageBreak
101      from 2019 to roll forward to that date!\MessageBreak
102      --- I'm giving up!\MessageBreak\MessageBreak
103      Note that manually loading the expl3 package\MessageBreak
104      from your distribution is not enough}%
105     \batchmode \read -1 to \reserved@a
106 </|latexrelease>
107     }%
108     {\input expl3.ltx }%
109   }
110 <|latexrelease> \EndIncludeInRelease
111 <|latexrelease>

```

To support roll-forward for the case where `xparse` is fully integrated into the kernel, we do not need to repeat the complex test above as we can simply look for the marker command.

```

112 <|latexrelease> \IncludeInRelease{2020/02/02}%
113 <|latexrelease>           {expl3}{Pre-load expl3}%

```

```

114 〈latexrelease〉\IfFileExists{expl3.ltx}{%
115 〈latexrelease〉  {%
116 〈latexrelease〉    \ifnum0%
117 〈latexrelease〉      \ifdefinable\pdffilesize 1\fi
118 〈latexrelease〉      \ifdefinable\filesize 1\fi
119 〈latexrelease〉      \ifdefinable\luatexversion\ifnum\luatexversion>94 1\fi\fi
120 〈latexrelease〉      >0 %
121 〈latexrelease〉    \else
122 〈latexrelease〉      \message{Skipping expl3-dependent extensions}
123 〈latexrelease〉      \expandafter\@gobbletwo
124 〈latexrelease〉    \fi
125 〈latexrelease〉  }
126 〈latexrelease〉  {%
127 〈latexrelease〉    \message{Skipping expl3-dependent extensions}%
128 〈latexrelease〉    \@gobbletwo
129 〈latexrelease〉  }%
130 〈latexrelease〉\input{expl3.ltx}
131 〈latexrelease〉\EndIncludeInRelease

```

Now in `\textrm{ latexrelease }` mode, redefine a few commands to avoid “already defined” errors.

```
132 〈latexrelease〉\@ifundefined{ExplSyntaxOff}{}{\@latexrelease@postltxpl}
```

1.2 Using `expl3` code

In order to ease the implementation of some new features in L^AT_EX 2 _{ε} we may (temporarily) use some coding based on the `expl3`-code. Such macros will eventually vanish and may be changed unannounced. They are there for internal use in the L^AT_EX 2 _{ε} kernel and are not meant to be used in third-party packages. These macros will always have the `@expl@` prefix in their name.

The rest of the name matches the `expl3` name but with all underscores replaced by @s and the : replaced by @@, e.g.,

```
\cs_new_eq:NN \@expl@tl@trim@spaces@apply@@nN \tl_trim_spaces_apply:nN
```

if that `expl3` command is needed in places that are others coded in L^AT_EX 2 _{ε} conventions.

In this file, each release of LaTeX adds an `\IncludeInRelease` block, in which the macros copied for that release were defined. In case a rollback is requested, the entire block is changed.

Each macro copied has a `\changes` entry to explain when and why it was copied, so that further to that may spot it easily.

Here `\cs_gset_eq:NN` is used, instead of the `new` variant because if different releases use that same name for different purposes, each can copy the macro without worrying about redefinitions.

```

133 〈latexrelease〉\IncludeInRelease{2020/10/01}{\@expl@cs@to@str@@N}%
134 〈latexrelease〉          {expl3 macros added for the 2020-10-01 release}%

```

The `expl3` activation needs to be inside the release guards as otherwise rolling forward is broken in old kernels that do not have `expl3` loaded.

```

135 \ExplSyntaxOn
136 \cs_gset_eq:NN \@expl@cs@to@str@@N \cs_to_str:N
137 \cs_gset_eq:NN \@expl@str@if@eq@nnTF \str_if_eq:nnTF

```

```

138 \cs_gset_eq:NN \expl@cs@prefix@spec@@N \cs_prefix_spec:N
139 \cs_gset_eq:NN \expl@cs@argument@spec@@N \cs_argument_spec:N
140 \cs_gset_eq:NN \expl@cs@replacement@spec@@N \cs_replacement_spec:N

141 \cs_gset_eq:NN \expl@str@map@function@@NN \str_map_function:NN
142 \cs_gset_eq:NN \expl@char@generate@@nn \char_generate:nn
143 \ExplSyntaxOff

144 <texrelease>\EndIncludeInRelease
145 <texrelease>\IncludeInRelease{0000/00/00}{\expl@cs@to@str@@N}%
146 <texrelease>      {expl3 macros added for the 2020-10-01 release}%
147 <texrelease>\let \expl@cs@to@str@@N \undefined
148 <texrelease>\let \expl@str@if@q@nnTF \undefined
149 <texrelease>\let \expl@cs@prefix@spec@@N \undefined
150 <texrelease>\let \expl@cs@argument@spec@@N \undefined
151 <texrelease>\let \expl@cs@replacement@spec@@N \undefined
152 <texrelease>\let \expl@str@map@function@@NN \undefined
153 <texrelease>\EndIncludeInRelease
154 </2ekernel | texrelease>

```

2 Document-level command names for `expl3` functions

Current home for L3 programming layer functions that we make directly available at the document level. This section may need to be moved later (after `\NewDocumentCommand` is defined in case we want to use that in the setup).

`\fpeval` The expandable command `\fpeval` takes as its argument a floating point expression and produces a result using the normal rules of mathematics. As this command is expandable it can be used where TeX requires a number and for example within a low-level `\edef` operation to give a purely numerical result. See `usrguide3` for further explanation.

`\inteval` The expandable command `\inteval` takes as its argument an integer expression and `\dimeval` produces a result using the normal rules of mathematics. The operations recognised are `\skipeval` +, -, *, and / plus parentheses. Division occurs with *rounding*, and ties are rounded away from zero. As this command is expandable it can be used where TeX requires a number and for example within a low-level `\edef` operation to give a purely numerical result. See `usrguide3` for further explanation. `\dimeval` and `\skipeval` are similar, but generate fixed and rubber length values, respectively.

`\fpeval` A document level wrapper around the code level function for floating point calculations.
`\inteval`
`\dimeval`
`\skipeval`

```

155 <2ekernel | texrelease>
156 <texrelease>\IncludeInRelease{2022/06/01}%
157 <texrelease>          {\fpeval}{fp and int calculations}%
158 \ExplSyntaxOn
159 \cs_new_eq:NN \fpeval \fp_eval:n

```

And a few more, this time wrappers around the eTeX primitives.

```

160 \cs_new_eq:NN \inteval \int_eval:n

```

```

161 \cs_new_eq:NN \dimeval  \dim_eval:n
162 \cs_new_eq:NN \skipEval \skip_eval:n
163 \ExplSyntaxOff

(End definition for \fpeval and others.)

164 ⟨/2ekernel | latexrelease⟩
165 ⟨latexrelease⟩\EndIncludeInRelease
166 ⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
167 ⟨latexrelease⟩          {\fpeval}{fp and int calculations}%
168 ⟨latexrelease⟩
169 ⟨latexrelease⟩\let\fpeval\@undefined
170 ⟨latexrelease⟩\let\inteval\@undefined
171 ⟨latexrelease⟩\let\dimeval\@undefined
172 ⟨latexrelease⟩\let\skipEval\@undefined
173 ⟨latexrelease⟩\EndIncludeInRelease

```

\UserName When declaring new commands with `\NewDocumentCommand` or `\NewCommandCopy` **\ExpandArgs** or similar, it is sometimes necessary to “construct” the csname. As a general mechanism the L3 programming layer has `\exp_args:N...` for this, but there is no mechanism for it if `\ExplSyntaxOn` is not active. We therefore offer a few of these commands also with CamelCase names.

\UserName A document wrapper for changing arguments to cs names for use with `\NewDocumentCommand` and similar functions.

```

174 ⟨*2ekernel | latexrelease⟩
175 ⟨latexrelease⟩\IncludeInRelease{2022/06/01}%
176 ⟨latexrelease⟩          {\ExpandArgs}{Some pre-expansion commands}%
177 \ExplSyntaxOn
178 \cs_new_eq:NN \UserName \use:c
179 \cs_new:Npn \ExpandArgs #1
180 {
181   \cs_if_exist_use:cF { \exp_args:N #1 }
182   { \msg_expandable_error:nnn { kernel } { unknown-arg-expansion } { #1 } }
183 }
184 \msg_new:nnn { kernel } { unknown-arg-expansion }
185   { Unknown-arg-expansion~"#1" }
186 \ExplSyntaxOff

```

(End definition for \UserName and \ExpandArgs.)

```

187 ⟨/2ekernel | latexrelease⟩
188 ⟨latexrelease⟩\EndIncludeInRelease
189 ⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
190 ⟨latexrelease⟩          {\ExpandArgs}{Some pre-expansion commands}%
191 ⟨latexrelease⟩
192 ⟨latexrelease⟩\let\UserName\@undefined
193 ⟨latexrelease⟩\let\ExpandArgs\@undefined
194 ⟨latexrelease⟩\EndIncludeInRelease

```

File f

ltdfns.dtx

1 Definitions

This section contains commands used in defining other macros.

1 `(*2ekernel)`

1.1 Initex initializations

`\two@digits` Prefix a number less than 10 with ‘0’.

2 `\def\two@digits#1{\ifnum#1<10 0\fi\number#1}`

(*End definition for \two@digits.*)

`\typeout` Display something on the terminal.

3 `</2ekernel>`
4 `<*2ekernel | latexrelease>`
5 `<latexrelease>\IncludeInRelease{2020/10/01}%`
6 `<latexrelease> \typeout{\allow "par" in \typeout}%`
7 `\protected\long\def\typeout#1{\begingroup`
8 `\set@display@protect`
9 `\def\par{\^J}%`
10 `\immediate\write\@unused{#1}\endgroup}`
11 `</2ekernel | latexrelease>`
12 `<latexrelease>\EndIncludeInRelease`
13 `<latexrelease>\IncludeInRelease{0000/00/00}%`
14 `<latexrelease> \typeout{\allow "par" in \typeout}%`
15 `<latexrelease>`
16 `<latexrelease>\def\typeout#1{\begingroup\set@display@protect`
17 `<latexrelease> \immediate\write\@unused{#1}\endgroup}`
18 `<latexrelease>\EndIncludeInRelease`
19 `(*2ekernel)`

(*End definition for \typeout.*)

`\newlinechar` A char to be used as new-line in output to files.

20 `\newlinechar`\^J`

(*End definition for \newlinechar.*)

1.2 Saved versions of TeX primitives

The TeX primitive `\foo` is saved as `\@@foo`. The following primitives are handled in this way:

`\@@par`

21 `\let\@@par=\par`
22 `%\let\@@input=\input %% moved earlier`
23 `%\let\@@end=\end %%`

(*End definition for \@@par.*)

\@@hyph Save original primitive definition.
²⁴ \let\@@hyph=\-
(End definition for \@@hyph.)

\@@italiccorr Save the original italic correction.
²⁵ \let\@@italiccorr=\/
(End definition for \@@italiccorr.)

\@height The following definitions save token space. E.g., using \@height instead of height saves
\@depth 5 tokens at the cost in time of one macro expansion.
\@width ²⁶ \def\@height{height} \def\@depth{depth} \def\@width{width}
\@minus ²⁷ \def\@minus{minus}
\@plus ²⁸ \def\@plus{plus}

The next one is another 100 tokens worth.
²⁹ \def\hb@xt@{\hbox to}
(End definition for \@height and others.)
³⁰ \message{hacks,}
\hb@xt@

1.3 Command definitions

This section defines the following commands:

\@namedef {\{NAME\}}
Expands to \def{\{NAME\}}, except name can contain any characters.

\@nameuse {\{NAME\}}
Expands to \{\{NAME\}\}.

\@ifnextchar X{\{YES\}}{\{NO\}}
Expands to {\{YES\}} if next character is an 'X', and to {\{NO\}} otherwise. (Uses \reserved@a-\reserved@c.) NOTE: GOBBLES ANY SPACE FOLLOWING IT.

\@ifstar {\{YES\}}{\{NO\}}
Gobbles following spaces and then tests if next the character is a '*'. If it is, then it gobbles the '*' and expands to {\{YES\}}, otherwise it expands to {\{NO\}}.

\@dblarg {\{CMD\}}{\{ARG\}}
Expands to \{\{CMD\}\}{\{ARG\}}{\{ARG\}}. Use \@dblarg\CS when \CS takes arguments [ARG1]{ARG2}, where default is ARG1 = ARG2.

\@ifundefined {\{NAME\}}{\{YES\}}{\{NO\}}
: If \NAME is undefined then it executes {\{YES\}}, otherwise it executes {\{NO\}}. More precisely, true if \NAME either undefined or = \relax.

\@ifdefinable {\{NAME\}}{\{YES\}} Executes {\{YES\}} if the user is allowed to define \NAME, otherwise it gives an error. The user can define \NAME if \@ifundefined{\NAME} is true, 'NAME' ≠ 'relax' and the first three letters of 'NAME' are not 'end', and if \endNAME is not defined.

\newcommand *{\{FOO\}}[{\{i\}}]{\{TEXT\}}
User command to define \FOO to be a macro with i arguments (i = 0 if missing) having the definition {\{TEXT\}}. Produces an error if \FOO already defined.

Normally the command is defined to be \long (ie it may take multiple paragraphs in its argument). In the star-form, the command is not defined as \long and a blank line in any argument to the command would generate an error.

\renewcommand *{\{FOO\}}[{\{i\}}]{\{TEXT\}}

```

Same as \newcommand, except it checks if \FOO already defined.
\newenvironment *{\{FOO\}}[\langle i\rangle]{\{DEF1\}}{\{DEF2\}}
equivalent to:
\newcommand{\FOO}[i]{\def{\endFOO}{DEF2}}
(or the appropriate star forms).

\renewenvironment
    Obvious companion to \newenvironment.

\@cons : See description of \output routine.
\@car \@car T1 T2 ... Tn\@nil == T1 (unexpanded)
\@cdr \@cdr T1 T2 ... Tn\@nil == T2 ... Tn (unexpanded)
\typeout {\langle message\rangle}
Produces a warning message on the terminal.

\typein {\langle message\rangle}
Types message, asks the user to type in a command, then executes it
\typein [(\CS)]{\MSG}
Same as above, except defines \CS to be the input instead of executing it.

\typein
 31 \def\typein{%
 32   \let\@typein\relax
 33   \@testopt\@xtypein\@typein}

 34 \ifx\directlua\undefined
 35 \def\@xtypein[#1]#2{%
 36   \typeout{#2}%
 37   \advance\endlinechar\@M
 38   \read\@inputcheck to#1%
 39   \advance\endlinechar-\@M
 40   \@typein}%

 41 \else
 42 \def\@xtypein[#1]#2{%
 43   \typeout{#2}%
 44   \begingroup \endlinechar\m@ne
 45   \read\@inputcheck to#1%
 46   \expandafter\endgroup
 47   \expandafter\def\expandafter#1\expandafter{#1}%
 48   \@typein}%

 49 \fi

(End definition for \typein.)

\@namedef
 50 \def\@namedef#1{\expandafter\def\csname #1\endcsname}

(End definition for \@namedef.)

\@nameuse
 51 \def\@nameuse#1{\csname #1\endcsname}

(End definition for \@nameuse.)

```

```

\@cons
52 \def\@cons#1#2{\begingroup\let\@elt\relax\xdef#1{#1\@elt #2}\endgroup}
(End definition for \@cons.)

\@car
\@cdr 53 \def\@car#1#2\@nil{#1}
54 \def\@cdr#1#2\@nil{#2}

(End definition for \@car and \@cdr.)

\@carcube \@carcube T1 ... Tn\@nil = T1 T2 T3 , n > 3
55 ⟨/2ekernel⟩
56 ⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\@carcube}{Make \@carcube long}%
57 ⟨*2ekernel | latexrelease⟩
58 \long\def\@carcube#1#2#3#4\@nil{#1#2#3}
59 ⟨/2ekernel | latexrelease⟩
60 ⟨latexrelease⟩\EndIncludeInRelease
61 %
62 ⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@carcube}{Undo: Make \@carcube long}%
63 ⟨latexrelease⟩\def\@carcube#1#2#3#4\@nil{#1#2#3}
64 ⟨latexrelease⟩\EndIncludeInRelease
65 ⟨*2ekernel⟩

(End definition for \@carcube.)

\@onlypreamble \@preamblecmds This macro adds its argument to the list of commands stored in \@preamblecmds to be disabled after \begin{document}. These commands are redefined to generate \@notprerr at this point.
66 \def\@preamblecmds{}
67 \def\@onlypreamble#1{%
68   \expandafter\gdef\expandafter\@preamblecmds\expandafter{%
69     \@preamblecmds\do#1}}
70 \@onlypreamble\@onlypreamble
71 \@onlypreamble\@preamblecmds

(End definition for \@onlypreamble and \@preamblecmds.)

\@star@or@long Look ahead for a *. If present reset \l@ngrel@x so that the next definition, #1, will be non-long.
72 \def\@star@or@long#1{%
73   \@ifstar
74   {\let\l@ngrel@x\relax#1}%
75   {\let\l@ngrel@x\long#1}}

(End definition for \@star@or@long.)

\l@ngrel@x This is either \relax or \long depending on whether the *-form of a definition command is being executed.
76 \let\l@ngrel@x\relax

(End definition for \l@ngrel@x.)

\newcommand User level \newcommand.
77 \def\newcommand{\@star@or@long\new@command}

```

```

\new@command 78 \def\new@command#1{%
79   @testopt{\@newcommand#1}0}

(End definition for \newcommand and \new@command.)

```

\@newcommand Handling arguments for \newcommand.
 \@argdef 80 \def\@newcommand#1[#2]{%
 \@xargdef 81 \kernel@ifnextchar [{\@argdef#1[#2]}{%
 82 {\@argdef#1[#2]}}}

Define #1 if it is definable.

Both here and in \@xargdef the replacement text is absorbed as an argument because if we are not allowed to make the definition we have to get rid of it completely.

```

83 \long\def\@argdef#1[#2]#3{%
84   \@ifdefinable #1{\@yargdef#1\@ne{#2}{#3}}}

```

Handle the second optional argument.

```

85 \long\def\@xargdef#1[#2][#3]#4{%
86   \@ifdefinable#1{%

```

Define the actual command to be:

```
\def\foo{\@protected@testopt\foo\\foo{default}}
```

where \\foo is a csname generated from applying \csname and \string to \foo, ie the actual name contains a backslash and therefore can't clash easily with existing command names. "Default" is the contents of the second optional argument of (re)newcommand.

```

87   \expandafter\def\expandafter#1\expandafter{%
88     \expandafter
89     \@protected@testopt
90     \expandafter
91     #1%
92     \csname\string#1\endcsname
93     {#3}}%

```

Now we define the internal macro ie \\foo which is supposed to pick up all arguments (optional and mandatory).

```

94   \expandafter\@yargdef
95     \csname\string#1\endcsname
96     \tw@
97     {#2}%
98     {#4}}}}

```

(End definition for \@newcommand, \@argdef, and \@xargdef.)

\@testopt This macro encapsulates the most common call to \@ifnextchar, saving several tokens each time it is used in the definition of a command with an optional argument. #1 The code to execute in the case that there is a [need not be a single token but can be any sequence of commands that 'expects' to be followed by [. If this command were only used in \newcommand definitions then #1 would be a single token and the braces could be omitted from {#1} in the definition below, saving a bit of memory.

```

99 \long\def\@testopt#1#2{%
100  \kernel@ifnextchar[{#1}{#1[{#2}]}}}

```

(End definition for \@testopt.)

\@protected@testopt Robust version of \@testopt. The extra argument (#1) must be a single token. If protection is needed the call expands to \protect applied to this token, and the 2nd and 3rd arguments are discarded (by \cx@protect). Otherwise \@testopt is called on the 2nd and 3rd arguments.

This method of making commands robust avoids the need for using up two csnames per command, the price is the extra expansion time for the \ifx test.

```

101 \def\@protected@testopt#1{%
102   \ifx\protect\@typeset@protect
103     \expandafter\@testopt
104   \else
105     \c@x@protect#1%
106   \fi}

```

(End definition for \@protected@testopt.)

\@yargdef These generate a primitive argument specification, from a L^AT_EX [*digit*] form; in fact *digit* can be anything such that \number*digit* is single digit.

Reorganised slightly so that \renewcommand{\reserved@a}[1]{foo} works. I am not sure this is worth it, as a following \newcommand would over-write the definition of \reserved@a.

Recall that L^AT_EX2.09 goes into an infinite loop with
\renewcommand[1]{\@tempa}{foo}
(DPC 6 October 93).

Reorganised again (DPC 1999). Rather than make a loop to construct the argument spec by counting, just extract the required argument spec by using a delimited argument (delimited by the digit). This is faster and uses less tokens. The coding is slightly odd to preserve the old interface (using #2 = \tw@ as the flag to surround the first argument with []). But the new method did not allow for the number of arguments #3 not being given as an explicit digit; hence (further expansion of this argument and use of) \number was added later in 1999.

It is not clear why these are still \long.

```

107 \long \def \@yargdef #1#2#3{%
108   \ifx#2\tw@
109     \def\reserved@b##1{####1}%
110   \else
111     \let\reserved@b\@gobble
112   \fi
113   \expandafter
114     \@yargd@f \expandafter{\number #3}#1%
115 }

116 \long \def \@yargd@f#1#2{%
117   \def \reserved@a ##1##2##{%
118     \expandafter\def\expandafter#2\reserved@b ##1##
119   }%
120   \l@ngrel@x \reserved@a 0##1##2##3##4##5##6##7##8##9##1%
121 }

```

(End definition for \@yargdef and \@yargd@f.)

\@reargdef

```

122 \long\def\@reargdef#1[#2]{%
123   \@yargdef#1\@ne{#2}}

```

(End definition for \creargdef.)

- \renewcommand Check the command name is already used. If not give an error message. Then temporarily disable \@ifdefinable then call \newcommand. (Previous version \let#1=\relax but this does not work too well if #1 is \tempa-e.)

124 \def\renewcommand{\@star@or@long\renew@command}

```
125 \def\renew@command#1{%
126   \begingroup \escapechar`m@ne\xdef\@gtempa{{\string#1}}\endgroup
127   \expandafter\@ifundefined\@gtempa
128     {\@latex@error{Command \string#1 undefined}\@ehc}%
129     \relax
130   \let\@ifdefinable\@rc@ifdefinable
131   \new@command#1}
```

(End definition for \renewcommand and \renew@command.)

- \@ifdefinable Test if user is allowed to define a command.

```
132 \long\def\@ifdefinable #1#2{%
133   \edef\reserved@a{\expandafter\@gobble\string #1}%
134   \@ifundefined\reserved@a
135     {\edef\reserved@b{\expandafter\@carcube \reserved@a xxx\@nil}%
136      \ifx \reserved@b\@qend \notdefinable\else
137        \ifx \reserved@a\@qrelax \notdefinable\else
138          #2%
139        \fi
140        \fi}%
141   \notdefinable}
```

Saved definition of \@ifdefinable.

142 \let\@ifdefinable\@ifdefinable

Version of \@ifdefinable for use with \renewcommand. Does not do the check this time, but restores the normal definition.

```
143 \long\def\@rc@ifdefinable#1#2{%
144   \let\@ifdefinable\@ifdefinable
145   #2}
```

(End definition for \@ifdefinable, @@ifdefinable, and \@rc@ifdefinable.)

- \newenvironment Define a new user environment. #1 is the environment name. #2# Grabs all the tokens up to the first {. These will be any optional arguments. They are not parsed at this point, but are just passed to \newenv which will eventually call \newcommand. Any optional arguments will then be parsed by \newcommand as it defines the command that executes the ‘begin code’ of the environment.

This #2# trick removed with version 1.2i as it fails if a { occurs in the optional argument. Now use \@ifnextchar directly.

146 \def\newenvironment{\@star@or@long\new@environment}

```
147 \def\new@environment#1{%
148   \@testopt{\newenva#1}{0}}
```

```

149 \def\@newenva#1[#2]{%
150     \kernel@ifnextchar [{\@newenvb#1[#2]}{\@newenv{#1}{[#2]}}}

151 \def\@newenvb#1[#2][#3]{\@newenv{#1}{[#2][[#3]]}}
(End definition for \newenvironment and others.)

\renewenvironment Redefine an environment. For \renewenvironment disable \ifdefinable and then call
\newenvironment. It is OK to \let the argument to \relax here as there should not
be a @temp... environment.
152 \def\renewenvironment{\@star@or@long\renew@environment}

\renew@environment
153 \def\renew@environment#1{%
154     \@ifundefined{#1}%
155         {\@latex@error{Environment #1 undefined}\@ehc
156             }\relax
157     \expandafter\let\csname#1\endcsname\relax
158     \expandafter\let\csname end#1\endcsname\relax
159     \new@environment{#1}}
(End definition for \renewenvironment and \renew@environment.)

\@newenv The internal version of \newenvironment.
Call \newcommand to define the <begin-code> for the environment. \def is used for
the <end-code> as it does not take arguments. (but may contain \pars)
Make sure that an attempt to define a ‘graf’ or ‘group’ environment fails.
160 \long\def\@newenv#1#2#3#4{%
161     \@ifundefined{#1}%
162         {\expandafter\let\csname#1\expandafter\endcsname
163             \csname end#1\endcsname}%
164     \relax
165     \expandafter\new@command
166         \csname #1\endcsname#2{#3}%
167         \l@ngrel@x\expandafter\def\csname end#1\endcsname{#4}{}}
(End definition for \@newenv.)

\newif And here’s a different sort of allocation: For example, \newif\iff foo creates \foottrue,
\foofalse to go with \iff foo.
168 \def\newif#1{%
169     \count@\escapechar \escapechar\m@ne
170     \let#1\iffalse
171     \@if#1\iftrue
172     \@if#1\iffalse
173     \escapechar\count@}

\@if
174 \def\@if#1#2{%
175     \expandafter\def\csname\expandafter\@gobbletwo\string#1%
176         \expandafter\@gobbletwo\string#2\endcsname
177         {\let#1#2}}

```

(End definition for \newif and \@if.)

\providecommand \providecommand takes the same arguments as \newcommand, but discards them if #1 is already defined. Otherwise it just acts like \newcommand. This implementation currently leaves any discarded definition in \reserved@a (and possibly \\reserved@a) this wastes a bit of space, but it will be reclaimed as soon as these scratch macros are redefined.

178 \def\providecommand{\@star@or@long\provide@command}

\provide@command 179 \def\provide@command#1{%
 180 \begingroup
 181 \escapechar\m@ne\xdef\@gtempa{{\string#1}}%
 182 \endgroup
 183 \expandafter\@ifundefined\@gtempa
 184 {\def\reserved@a{\new@command#1}}%
 185 {\def\reserved@a{\renew@command\reserved@a}}%
 186 \reserved@a}%

(End definition for \providecommand and \provide@command.)

\CheckCommand \CheckCommand takes the same arguments as \newcommand. If the command already exists, with the same definition, then nothing happens, otherwise a warning is issued. Useful for checking the current state before a macro package starts redefining things. Currently two macros are considered to have the same definition if they are the same except for different default arguments. That is, if the old definition was: \newcommand\xxx[2][a]{(#1)(#2)} then \CheckCommand\xxx[2][b]{(#1)(#2)} would not generate a warning, but, for instance \CheckCommand\xxx[2]{(#1)(#2)} would.

187 \def\CheckCommand{\@star@or@long\check@command}

\CheckCommand is only available in the preamble part of the document.

188 \onlypreamble\CheckCommand

\check@command 189 \def\check@command#1#2{\@check@c#1{#2}}
190 \onlypreamble\check@command

(End definition for \CheckCommand and \check@command.)

\@check@c \CheckCommand itself just grabs all the arguments we need, without actually looking for [optional argument forms. Now define \reserved@a. If \\reserved@a is then defined, compare it with the “\#1’ otherwise compare \reserved@a with #1.

191 \long\def\@check@c#1#2#3{%
 192 \expandafter\let\csname\string\reserved@a\endcsname\relax
 193 \renew@command\reserved@a#2{#3}%
 194 \@ifundefined{\string\reserved@a}%
 195 {\@check@eq#1\reserved@a}%
 196 {\expandafter\@check@eq
 197 \csname\string#1\expandafter\endcsname
 198 \csname\string\reserved@a\endcsname}%
 199 \onlypreamble\@check@c

(End definition for \@check@c.)

\@check@eq Complain if #1 and #2 are not \ifx equal.

```

200 \def \@check@eq#1#2{%
201   \ifx#1#2\else
202     \@latex@warning@no@line
203       {Command \noexpand#1 has
204        changed.\MessageBreak
205        Check if current package is valid}%
206   \fi}
207 \onlypreamble \@check@eq
```

(End definition for \@check@eq.)

\@gobble The \@gobble macro is used to get rid of its argument.
 \@gobbletwo
 \@gobblethree
 \@gobblefour

```

208 \long\def \@gobble #1{%
209   \long\def \@gobbletwo #1#2{%
210     \long\def \@gobblethree #1#2#3{%
211       \long\def \@gobblefour #1#2#3#4{}}
```

(End definition for \@gobble and others.)

\@firstofone Some argument-grabbers.
 \@firstoftwo
 \@secondoftwo

```

212 \long\def \@firstofone#1{#1}
213 \long\def \@firstoftwo#1#2{#1}
214 \long\def \@secondoftwo#1#2{#2}
```

\@iden is another name for \@firstofone for compatibility reasons.

```
215 \let \@iden \@firstofone
```

(End definition for \@firstofone and others.)

\@thirdofthree Another grabber now used in the encoding specific section.

```

216 \long\def \@thirdofthree#1#2#3{#3}
```

(End definition for \@thirdofthree.)

\@expandtwoargs A macro to totally expand two arguments to another macro

```

217 \def \@expandtwoargs#1#2#3{%
218   \edef\reserved@a{\noexpand#1{#2}{#3}}\reserved@a}
```

(End definition for \@expandtwoargs.)

\@backslashchar A category code 12 backslash.

```

219 \edef \@backslashchar{\expandafter\@gobble\string\\}
```

(End definition for \@backslashchar.)

1.4 Robust commands and protect

Fragile and robust commands are one of the thornier issues in L^AT_EX's commands. Whilst typesetting documents, L^AT_EX makes use of many of T_EX's features, such as arithmetic, defining macros, and setting variables. However, there are (at least) three different occasions when these commands are not safe. These are called 'moving arguments' by L^AT_EX, and consist of:

- writing information to a file, such as indexes or tables of contents.
- writing information to the screen.
- inside an `\edef`, `\message`, `\mark`, or other command which evaluates its argument fully.

The method L^AT_EX uses for making fragile commands robust is to precede them with `\protect`. This can have one of four possible values:

- `\relax`, for normal typesetting. So `\protect\foo` will execute `\foo`.
- `\string`, for writing to the screen. So `\protect\foo` will write `\foo`.
- `\noexpand`, for writing to a file. So `\protect\foo` will write `\foo` followed by a space.
- `\@unexpandable@protect`, for writing a moving argument to a file. So `\protect\foo` will write `\protect\foo` followed by a space. This value is also used inside `\edefs`, `\marks` and other commands which evaluate their arguments fully. More precisely, whenever the content of an `\edef` or `\xdef` etc. can contain arbitrary user input not under the direct control of the programmer, one should use `\protected@edef` instead of `\edef`, etc., so that `\protect` has a suitable definition and the user input will not break if it contains fragile commands.

```
\@unexpandable@protect
 220 \def\@unexpandable@protect{\noexpand\protect\noexpand}
(End definition for \@unexpandable@protect.)
```

`\DeclareRobustCommand` `\declare@robustcommand` This is a package-writers command, which has the same syntax as `\newcommand`, but which declares a protected command. It does this by having

`\DeclareRobustCommand\foo`
define `\foo` to be `\protect\foo<space>`,
and then use `\newcommand\foo<space>`.

Since the internal command is `\foo<space>`, when it is written to an auxiliary file, it will appear as `\foo`.

We have to be a bit cleverer if we're defining a short command, such as `_`, in order to make sure that the auxiliary file does not include a space after the command, since `_ a` and `_a` aren't the same. In this case we define `_` to be:

```
\x@protect\_ \protect\_<space>
```

which expands to:

```
\ifx\protect\@typeset@protect\else
  \x@protect@_
\fi
\protect\_<space>
```

Then if `\protect` is `\@typeset@protect` (normally `\relax`) then we just perform `_<space>`, and otherwise `\x@protect@` gobbles everything up and expands to `\protect_`.

Note: setting `\protect` to any value other than `\relax` whilst in ‘typesetting’ mode will cause commands to go into an infinite loop! In particular, setting `\protect` to `\empty` will cause `_` to loop forever. It will also break lots of other things, such as protected `\ifmmode`s inside `\haligns`. If you really have to do such a thing, then please set `\@typeset@protect` to be `\empty` as well. (This is what the code for `\patterns` does, for example.)

More fun with `\expandafter` and `\csname`.

```
221 \def\DeclareRobustCommand{\@star@or@long\declare@robustcommand}
222 \def\declare@robustcommand#1{%
223   \ifx#1\undefined\else\ifx#1\relax\else
224     \x@info{Redefining \string#1}%
225   \fi\fi
226   \edef\reserved@a{\string#1}%
227   \def\reserved@b{%
228     \expandafter\strip@prefix\meaning\reserved@b}%
229   \edef#1{%
230     \ifx\reserved@a\reserved@b
231       \noexpand\x@protect
232       \noexpand#1%
233     \fi
234     \noexpand\protect
235     \expandafter\noexpand\csname
236     \expandafter\gobble\string#1 \endcsname
237   }%
238   \let\@ifdefinable\@rc@ifdefinable
239   \expandafter\new@command\csname
240     \expandafter\gobble\string#1 \endcsname
241 }
```

(End definition for `\DeclareRobustCommand` and `\declare@robustcommand`.)

```
\x@protect
\x@protect
242 \def\x@protect#1{%
243   \ifx\protect\@typeset@protect\else
244     \x@protect#1%
245   \fi
246 }
247 \def\x@protect#1\fi#2#3{%
248   \fi\protect#1%
249 }
```

(End definition for `\x@protect` and `\x@protect`.)

`\@typeset@protect` We set `\@typeset@protect` to `\relax` rather than `\empty` to make sure that the protection mechanism stops the look-ahead and expansion performed at the start of `\halign` cells.

```
250 \let\@typeset@protect\relax
```

(End definition for `\@typeset@protect`.)

`\set@display@protect` These macros set `\protect` appropriately for typesetting or displaying.

`\set@typeset@protect` 251 `\def\set@display@protect{\let\protect\string}`
252 `\def\set@typeset@protect{\let\protect\@typeset@protect}`

(End definition for `\set@display@protect` and `\set@typeset@protect`.)

`\protected@edef` The commands `\protected@edef` and `\protected@xdef` perform ‘safe’ `\edefs` and
`\xdefs`, saving and restoring `\protect` appropriately. For cases where restoring `\protect`
`\unrestored@protected@xdef` doesn’t matter, there’s an ‘unsafe’ `\unrestored@protected@xdef`, useful if you know
`\restore@protect` what you’re doing!

253 `\def\protected@edef{%`
254 `\let\@@protect\protect`
255 `\let\protect\@unexpandable@protect`
256 `\afterassignment\restore@protect`
257 `\edef`
258 `}`
259 `\def\protected@xdef{%`
260 `\let\@@protect\protect`
261 `\let\protect\@unexpandable@protect`
262 `\afterassignment\restore@protect`
263 `\xdef`
264 `}`
265 `\def\unrestored@protected@xdef{%`
266 `\let\protect\@unexpandable@protect`
267 `\xdef`
268 `}`
269 `\def\restore@protect{\let\protect\@@protect}`

(End definition for `\protected@edef` and others.)

`\protect` The normal meaning of `\protect`

270 `\set@typeset@protect`

(End definition for `\protect`.)

`\MakeRobust` This macro makes an existing fragile macro robust, but only if it hasn’t been robust
in the past, i.e., it checks for the existence of the macro `\<name>U` and if that exists it
assumes that `\<name>` is already robust. In that case either undefine the inner macro first
or use `\DeclareRobustCommand` to define it in a robust way directly. We could probably
test the top-level definition to have the right kind of structure, but this is somewhat
problematical as we then have to distinguish between `\long` macros and others and also
take into account that sometimes the top-level is deliberately done manually (like with
`\begin`).

The macro firstly checks if the control sequence in question exists at all.

271 `(/2ekernel)`
272 `(latexrelease)\IncludeInRelease{2020/10/01}{\MakeRobust}{\MakeRobust}%`
273 `(*2ekernel | latexrelease)`
274 `\def\MakeRobust#1{%`
275 `\count@=\escapechar`
276 `\escapechar='\\`
277 `\@ifundefined{\expandafter\@gobble\string#1}{%`
278 `\@latex@error{Command '\string#1' undefined.}%`

```

279      \MessageBreak There is nothing here to make robust}%
280      \@eha
281 }%

```

Then we check if the macro is already robust. We do this by testing if the internal name for a robust macro is defined, namely `\foo_`. If it is already defined do nothing, otherwise set `\foo_` equal to `\foo` and redefine `\foo` so that it acts like a macro defined with `\DeclareRobustCommand`. We use `\@kernel@rename@newcommand` to copy `\foo` over to `\foo_`, including a possible default optional argument.

```

282 }%
283   \@ifundefined{\expandafter\gobble\string#1\space}%
284 {%
285   \expandafter\@kernel@rename@newcommand
286     \csname\expandafter\gobble\string#1\space\endcsname
287     #1%
288   \edef\reserved@a{\string#1}%
289   \def\reserved@b{#1}%
290   \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
291   \xdef#1{%
292     \ifx\reserved@a\reserved@b
293       \noexpand\x@protect\noexpand#1%
294     \fi
295     \noexpand\protect\expandafter\noexpand
296     \csname\expandafter\gobble\string#1\space\endcsname}%
297   }%
298   {\@latex@info{Command ‘\string#1’ is already robust}}%
299 }%
300 \escapechar=\count@
301 }%

```

This macro renames a command, possibly with an optional argument (defined with `\newcommand`) from #2 to #1, by renaming the internal macro `\#2` to `\#1` and defining #1 appropriately, then undefining #2 and `\#2`. The `\afterassignment` trick is to make both definitions in `\copy@newcommand` global (which are local by default).

In case the macro was defined with `\newcommand` and an optional argument, to replicate exactly the behaviour of `\DeclareRobustCommand` we have to move also the internal `\foo` to `\foo_`. In that case, #1 will be a parameterless macro (`\robust@command@chk@safe` checks that), and `\@if@newcommand` will return true (both defined below in this file). If so, we can use `\copy@newcommand` rather than plain `\let` to copy the command over. `\@kernel@rename@newcommand` does this test and carries out the renaming.

```

302 \def\@kernel@rename@newcommand#1#2{%
303   \robust@command@chk@safe#2%
304   {\@if@newcommand#2%
305     {\afterassignment\global
306       \global\copy@newcommand#1#2%
307       \global\let#2@\undefined
308       \global\expandafter\let\csname\string#2\endcsname\@undefined}%
309     {\global\let#1=#2}%
310     {\global\let#1=#2}%
311   </2ekernel | latexrelease>
312   <| latexrelease>\EndIncludeInRelease

```

```

313 %
314 <|latexrelease>\IncludeInRelease{2019/10/01}{\MakeRobust}{\MakeRobust}%
315 <|latexrelease>\def\MakeRobust#1{%
316 <|latexrelease> \@ifundefined{\expandafter\gobble\string#1}{%
317 <|latexrelease> \@latex@error{The control sequence ‘\string#1’ is undefined!}%
318 <|latexrelease> \MessageBreak There is nothing here to make robust}%
319 <|latexrelease> \@eha
320 <|latexrelease> }%
321 <|latexrelease> {%
322 <|latexrelease> \@ifundefined{\expandafter\gobble\string#1\space}{%
323 <|latexrelease> \global\expandafter\let\csname
324 <|latexrelease> \expandafter\gobble\string#1\space\endcsname=\#1%
325 <|latexrelease> \edef\reserved@a{\string#1}%
326 <|latexrelease> \def\reserved@b{\#1}%
327 <|latexrelease> \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
328 <|latexrelease> \xdef#1{%
329 <|latexrelease> \ifx\reserved@a\reserved@b
330 <|latexrelease> \noexpand\x@protect\noexpand#1%
331 <|latexrelease> \fi
332 <|latexrelease> \noexpand\protect\expandafter\noexpand
333 <|latexrelease> \csname\expandafter\gobble\string#1\space\endcsname}%
334 <|latexrelease> }%
335 <|latexrelease> {%
336 <|latexrelease> {\@latex@info{The control sequence ‘\string#1’ is already robust}}%
337 <|latexrelease> }%
338 <|latexrelease>}%
339 <|latexrelease>\let\@kernel@rename@newcommand\@undefined
340 <|latexrelease>\EndIncludeInRelease
341 %
342 <|latexrelease>\IncludeInRelease{2015/01/01}{\MakeRobust}{\MakeRobust}%
343 <|latexrelease>\def\MakeRobust#1{%
344 <|latexrelease> \@ifundefined{\expandafter\gobble\string#1}{%
345 <|latexrelease> \@latex@error{The control sequence ‘\string#1’ is undefined!}%
346 <|latexrelease> \MessageBreak There is nothing here to make robust}%
347 <|latexrelease> \@eha
348 <|latexrelease> }%
349 <|latexrelease> {%
350 <|latexrelease> \@ifundefined{\expandafter\gobble\string#1\space}{%
351 <|latexrelease> \expandafter\let\csname
352 <|latexrelease> \expandafter\gobble\string#1\space\endcsname=\#1%
353 <|latexrelease> \edef\reserved@a{\string#1}%
354 <|latexrelease> \def\reserved@b{\#1}%
355 <|latexrelease> \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
356 <|latexrelease> \xdef#1{%
357 <|latexrelease> \ifx\reserved@a\reserved@b
358 <|latexrelease> \noexpand\x@protect\noexpand#1%
359 <|latexrelease> \fi
360 <|latexrelease> \noexpand\protect\expandafter\noexpand
361 <|latexrelease> \csname\expandafter\gobble\string#1\space\endcsname}%
362 <|latexrelease> }%
363 <|latexrelease> {%
364 <|latexrelease> {\@latex@info{The control sequence ‘\string#1’ is already robust}}%
365 <|latexrelease> }%
366 <|latexrelease>}%

```

```

367 〈\latexrelease〉\let\@kernel@rename@newcommand\@undefined
368 〈\latexrelease〉\EndIncludeInRelease
369 %
370 〈\latexrelease〉\IncludeInRelease{0000/00/00}{\MakeRobust}{\MakeRobust}%
371 〈\latexrelease〉\let\MakeRobust\@undefined
372 〈\latexrelease〉\let\@kernel@rename@newcommand\@undefined
373 〈\latexrelease〉\EndIncludeInRelease
374 {*2ekernel}

```

(End definition for \MakeRobust and \@kernel@rename@newcommand.)

\kernel@make@fragile The opposite of \MakeRobust except that it doesn't do many checks as it is internal to the kernel. Why does one want such a thing? Only for compatibility reasons if \latexrelease requests a rollback of the kernel. For this reason we pretend that this command existed in all earlier versions of L^AT_EX i.e., we are not rolling it back since we need it precisely then. But we have to get it into the \latexrelease file so that a roll forward is possible too.

```

375 〈/2ekernel〉
376 {*2ekernel | \latexrelease}
377 〈\latexrelease〉\IncludeInRelease{2020/10/01}%
378 〈\latexrelease〉{\@kernel@make@fragile}{Undo robustness}%
379 \def\@kernel@make@fragile#1{%
380   \@ifundefined{\expandafter\gobble\string#1\space}%

```

If not robust do nothing.

```
381   {}%
```

Otherwise copy \foo_ back to \foo. Then use \@kernel@rename@newcommand to check and copy \\foo_ back to \\foo in case the command has an optional argument. If so, also undefine \\foo_, and at the end undefine \foo_.

```

382   {%
383     \global\expandafter\let\expandafter #1\csname
384       \expandafter\gobble\string#1\space\endcsname
385     \expandafter\@kernel@rename@newcommand
386       \csname\expandafter\gobble\string#1\expandafter\endcsname
387       \csname\expandafter\gobble\string#1\space\endcsname
388     \global\expandafter\let\csname
389       \expandafter\gobble\string#1\space\endcsname\@undefined
390   }%
391 }

392 〈\latexrelease〉\EndIncludeInRelease
393 %
394 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
395 〈\latexrelease〉{\@kernel@make@fragile}{Undo robustness}%
396 \def\@kernel@make@fragile#1{%
397   \ifundefined{\expandafter\gobble\string#1\space}%
398   {}%
399   {}%
400   \global\expandafter\let\expandafter #1\csname
401   \expandafter\gobble\string#1\space\endcsname
402   \global\expandafter\let\csname
403   \expandafter\gobble\string#1\space\endcsname\@undefined
404   {}%
405 }
```

```

406 〈/latexrelease〉\EndIncludeInRelease
407 〈/2ekernel | latexrelease〉
408 〈*2ekernel〉

(End definition for \kernel@make@fragile.)

```

1.5 Acting on robust commands

```

409 〈/2ekernel〉
410 〈/latexrelease〉\IncludeInRelease{2020-10-01}{\robust@command@act}
411 〈/latexrelease〉 {Add \robust@command@act}%
412 〈*2ekernel | latexrelease〉

```

With most document level commands being robust now there is more of a requirement to have a standard way of aliasing (or copying) a command to a new name, for example to save an original definition before changing a command. `\DeclareCommandCopy` is analogous to `TEX`'s `\let`, except that it copes with the different types of robust commands defined by L^AT_EX's mechanisms.

A couple of “types of robustness” are defined by the L^AT_EX 2 _{ε} kernel, namely robust commands defined with `\DeclareRobustCommand` and commands with optional arguments defined with `\newcommand`. However there are other types of robust commands that are frequently used, which are not defined in the L^AT_EX 2 _{ε} kernel, like commands defined with `xparse`'s `\NewDocumentCommand` and `etoolbox`'s `\newrobustcmd`.

In this section we will define a generic extensible machinery to act on robust commands. This code will then be used to test if a command is robust, considered the different types of robustness, and then either copy that definition, if `\DeclareCommandCopy` (or similar) is used, or show the definition of the command, if `\ShowCommand` is used.

`\robust@command@act` The looping machinery is generic and knows nothing about what is to be done for each case. The syntax of the main macro `\robust@command@act` is:

```
\robust@command@act<action-list><robust-cmd>
  <fallback-action><act-arg>
```

`<action-list>` is a token list of the form:

```
{<if-type-1> <act-type-1>}
 {<if-type-2> <act-type-2>}
 ...

```

`\robust@command@act` will iterate over the `<action-list>`, evaluating each `<if-type-n> <robust-cmd> {<true>} {<false>}`. If the `<if-type-n>` conditional returns `<true>`, then `<act-type-n><act-arg>` is executed, and the loop ends. If the conditional returns `<false>`, then `<if-type-n+1>` is executed in the same way, until either one of the conditionals return `<true>`, or the end of the `<action-list>` is reached. If the end is reached, then `<fallback-action><act-arg>` is executed before `\robust@command@act` exits.

`\robust@command@act` will start by using `\robust@command@act@chk@args` to check if the `<robust-cmd>` (#2) is a parameterless (possibly `\protected`) macro. If it is not, the command is not a robust command: these always start with a parameterless user-level macro; in that case, `\robust@command@act@end` is used to short-circuit the process and do the `<fallback-action>` (#3). This first test is necessary because later on we need to be able to expand the `<robust-cmd>` without the risk of it Breaking Badly, and as a bonus, this speeds up the process in case we used `\NewCommandCopy` in a “normal” macro.

```
413 \long\def\robust@command@act#1#2#3#{%
```

```

414   \robust@command@chk@safe#2%
415   {\expandafter\robust@command@act@loop
416     \expandafter#2%
417     #1{\@nnil\@nnil}%
418     \robust@command@act@end}%
419   {\robust@command@act@end}%
420   {#3}{#4}}%

```

If `\robust@command@act@chk@args` branched to false, then `\robust@command@act@loop` will loop over the list of items in the *<action-list>* (#1), and process each item as described earlier. If the *<if-type-n>* command expands to *<true>* then `\robust@command@act@do` is used to execute *<act-type-n>* on the *<act-arg>*, otherwise the loop resumes with the next item.

```

421 \long\def\robust@command@act@loop#1#2{\robust@command@act@loop@aux#1#2}
422 \long\def\robust@command@act@loop@aux#1#2#3{%
423   \ifx\@nnil#2%
424   \else
425     #2{#1}%
426     {\robust@command@act@do{#3}}%
427     {\expandafter\robust@command@act@loop\expandafter#1}%
428   \fi}
429 \long\def\robust@command@act@do#1%
430   \fi#2%
431   \robust@command@act@end#3#4{%
432   \fi
433   #1#4}

```

If the end is reached and no action was taken, then do *<fallback-action>**<act-arg>*.

```
434 \long\def\robust@command@act@end#1#2{#1#2}
```

```

\robust@command@chk@safe
\robust@command@act@chk@args
435 \long\def\robust@command@chk@safe#1{%
436   \begingroup
437   \escapechar='\\
438   \expandafter\endgroup\expandafter
439   \robust@command@act@chk@args\meaning#1:->\@nil}%
440 \def\robust@command@act@chk@args#1:->#2\@nil{%
441   \@expl@str@if@eq@@nnTF{#1}{macro}%
442   {\@firstoftwo}%
443   {\@expl@str@if@eq@@nnTF{#1}{\protected macro}%
444   {\@firstoftwo}%
445   {\@secondoftwo}}}}

446 </2ekernel | latexrelease>
447 <latexrelease>\EndIncludeInRelease
448 <latexrelease>\IncludeInRelease{0000-00-00}{\robust@command@act}
449 <latexrelease> {Add \robust@command@act}%
450 <latexrelease>\let\robust@command@act@\undefined
451 <latexrelease>\let\robust@command@act@loop@\undefined
452 <latexrelease>\let\robust@command@act@loop@aux@\undefined
453 <latexrelease>\let\robust@command@act@do@\undefined
454 <latexrelease>\let\robust@command@act@end@\undefined
455 <latexrelease>\let\robust@command@chk@safe@\undefined
456 <latexrelease>\let\robust@command@act@chk@args@\undefined

```

```

457  ⟨{latexrelease}⟩\EndIncludeInRelease
458  ⟨*2ekernel⟩

(End definition for \robust@command@act and others.)

```

1.5.1 Copying robust commands

```

459  ⟨/2ekernel⟩
460  ⟨{latexrelease}⟩\IncludeInRelease[2020-10-01]{\DeclareCommandCopy}
461  ⟨{latexrelease}⟩ {Add \NewCommandCopy, \RenewCommandCopy, and \DeclareCommandCopy}%
462  ⟨*2ekernel | latexrelease⟩

\NewCommandCopy \RenewCommandCopy \DeclareCommandCopy

```

\NewCommandCopy starts by checking if #1 is already defined, and raises an error if so, otherwise the definition is carried out. \RenewCommandCopy does (almost) the opposite. If the command is *not* defined, then an error is raised. But the definition is carried out anyhow, so the behaviour is consistent with \renewcommand.

A \ProvideCommandCopy isn't defined because it's not reasonably useful. \provide... commands mean "define this if there's no other definition", but copying a command (usually) implies that the command being copied is defined, so \ProvideCommandCopy doesn't make a lot of sense. But more importantly, the most common use case of copying a command is to redefine it later, while preserving the old definition, as in:

```

\ProvideCommandCopy \A \B
\renewcommand \B { ... \A ... }

```

then, if \A is already defined the first line is skipped, an in this case \B won't work as expected.

The three versions call the internal \declare@commandcopy with the proper action. \@firstofone will carry out the copy. The only case when the copy is not made is the ⟨false⟩ case for \NewCommandCopy, in which the command already exists and the definition is aborted.

```

463  \def\NewCommandCopy{%
464    \declare@commandcopy
465    {\@firstofone}%
466    {\@firstoftwo\@notdefinable}%
467  \def\RenewCommandCopy{%
468    \declare@commandcopy
469    {\@latex@error{Command \backslash reserved@a\space undefined}\@ehc
470     \@firstofone}%
471    {\@firstofone}%
472  \def\DeclareCommandCopy{%
473    \declare@commandcopy
474    {\@firstofone}%
475    {\@firstofone}}%

```

Start by checking if the command is already defined. The proper action is taken by each specific command above. If all's good, then \robust@command@act is called with the proper arguments as described earlier, with \declarecommandcopylisthook as the ⟨action-list⟩ and \declare@commandcopy@let as the ⟨fallback-action⟩.

```

476  \long\def\declare@commandcopy#1#2#3#4{%
477    \edef\reserved@a{\@expl@cs@to@str@ON#3}%
478    \@ifundefined\reserved@a{#1}{#2}%
479    {\robust@command@act
480      \declarecommandcopylisthook#4%
481      \declare@commandcopy@let{#3#4}}}

```

The initial definition of `\@declarecommandcopylisthook` contains the tests for the two types of robust command in the kernel.

```
482 \def\@declarecommandcopylisthook{%
483   {\@if@DeclareRobustCommand \copy@DeclareRobustCommand}%
484   {\@if@newcommand \copy@newcommand}}
```

The initial definition of `\@declarecommandcopylisthook` contains the tests for the two types of robust command in the kernel.

```
485 \long\def\declare@commandcopy@let{\let#1=\relax}
```

`\declare@commandcopy@let`

Now the rollback code.

```
486 {/2ekernel | latexrelease}
487 {\latexrelease}\EndIncludeInRelease
488 {\latexrelease}\IncludeInRelease{0000-00-00}{\DeclareCommandCopy}
489 {\latexrelease} {Undefined \NewCommandCopy, \RenewCommandCopy, and \DeclareCommandCopy}%
490 {\latexrelease}\let\NewCommandCopy\undefined
491 {\latexrelease}\let\RenewCommandCopy\undefined
492 {\latexrelease}\let\DeclareCommandCopy\undefined
493 {\latexrelease}\let\declare@commandcopy\undefined
494 {\latexrelease}\let\@declarecommandcopylisthook\undefined
495 {\latexrelease}\let\declare@commandcopy@let\undefined
496 {\latexrelease}\EndIncludeInRelease
497 {*2ekernel}
```

(*End definition for \NewCommandCopy and others.*)

1.5.2 Showing robust commands

`\ShowCommand` Most of the machinery defined for `\NewCommandCopy` can be used to show the definition of a robust command, in a similar fashion to `\texdef`. The difference is that after the command's is detected to has a given type of robustness, rather than making a copy, we use a separate routine to show its definition.

With all the machinery in place, `\ShowCommand` itself is quite simple: use `\robust@command@act` to iterate through the `\@showcommandlisthook` list, and if nothing is found, fallback to `\show`.

```
498 {/2ekernel}
499 {\latexrelease}\IncludeInRelease{2020-10-01}{\ShowCommand}%
500 {\latexrelease} {Add \ShowCommand}%
501 {*2ekernel | latexrelease}

502 \long\def\ShowCommand#1{%
503   \robust@command@act
504   {\@showcommandlisthook#1}%
505   \show#1}
```

The initial definition of `\@showcommandlisthook` contains the same tests as used for copying, but `\@show@...` commands instead of `\copy@....` Same as before, it is initialized to cope with `\DeclareRobustCommand` and `\newcommand` with optional arguments.

```
506 \def\@showcommandlisthook{%
507   {\@if@DeclareRobustCommand \show@DeclareRobustCommand}%
508   {\@if@newcommand \show@newcommand}}
```

Now the rollback code.

```
509 </2ekernel | latexrelease>
510 <latexrelease>\EndIncludeInRelease
511 <latexrelease>\IncludeInRelease{0000-00-00}{\ShowCommand}
512 <latexrelease> {Undefined \ShowCommand}%
513 <latexrelease>\let\ShowCommand\@undefined
514 <latexrelease>\let\@showcommandlisthook\@undefined
515 <latexrelease>\EndIncludeInRelease
516 <*2ekernel>

(End definition for \ShowCommand and \@showcommandlisthook.)
```

```
517 </2ekernel>
518 <latexrelease>\IncludeInRelease{2020-10-01}{\@if@DeclareRobustCommand}
519 <latexrelease> {Add \@if@DeclareRobustCommand, \@if@newcommand,
520 <latexrelease> \copy@DeclareRobustCommand, \copy@newcommand,
521 <latexrelease> \show@DeclareRobustCommand, \show@newcommand}%
522 <*2ekernel | latexrelease>
```

1.5.3 Commands defined with \DeclareRobustCommand

\@if@DeclareRobustCommand Now that we provided a generic way to copy one macro to another, we need to define a way to check if a command is one of L^AT_EX 2_ε's robust types. These tests are heavily based on Heiko's \LetLtxMacro, but chopped into separate macros.

\@if@DeclareRobustCommand checks if a command \cmd was defined by \DeclareRobustCommand. The test returns true if the expansion of \cmd is exactly \protect\cmd.

```
523 \long\def\@if@DeclareRobustCommand#1{%
524   \begingroup
525   \escapechar='\
526   \edef\reserved@a{\string#1}%
527   \edef\reserved@b{\detokenize{#1}}%
528   \xdef\@gtempa{%
529     \ifx\reserved@a\reserved@b
530       \noexpand\x@protect
531       \noexpand#1%
532     \fi
533     \noexpand\protect
534     \expandafter\noexpand\csname\expl@cs@to@str@N#1 \endcsname}%
535   \endgroup
536   \ifx\@gtempa#1\relax
537     \expandafter\@firstoftwo
538   \else
539     \expandafter\@secondoftwo
540   \fi}
```

If a command was defined by \DeclareRobustCommand (that is, \@if@DeclareRobustCommand returns true), then to make a copy of \cmd into \foo we define the latter such that it expands to \protect\foo, then make \foo equal to \cmd.

There is one detail we need to take care of: if a command was defined with \DeclareRobustCommand it may still have an optional argument, in which case there is one more macro layer before the actual definition of the command. We use \@if@newcommand to check that and \copy@newcommand to do the copying.

```
541 \long\def\copy@DeclareRobustCommand#1#2{%
```

```

542 \begingroup
543   \escapechar='\\
544   \edef\reserved@a{\string#1}%
545   \edef\reserved@b{\detokenize{#1}}%
546   \edef\reserved@a{%
547 \endgroup
548 \def\noexpand#1{%
549   \ifx\reserved@a\reserved@b
550     \noexpand\x@protect
551     \noexpand#1%
552   \fi
553   \noexpand\protect
554   \expandafter\noexpand\csname\@expl@cs@to@str@@N#1 \endcsname}%
555 \noexpand\copy@kernel@robust@command
556   \expandafter\noexpand\csname\@expl@cs@to@str@@N#1 \endcsname
557   \expandafter\noexpand\csname\@expl@cs@to@str@@N#2 \endcsname}%
558 \reserved@a}
559 \long\def\copy@kernel@robust@command#1#2{%
560   \robust@command@chk@safe#2%
561   {\@if@newcommand#2%
562     {\@copy@newcommand}%
563     {\declare@commandcopy@let}%
564     {\declare@commandcopy@let}%
565   #1#2}

```

Showing the command is pretty simple. This command prints the top-level expansion as TeX's `\show` would, but with `robust macro:` rather than just `macro:`, then a blank line and then `\show` the inner command. For a macro defined with, say, `\DeclareRobustCommand\foo[1]{bar}`, it will print:

```

> \foo=robust macro:
->\protect \foo . 

> \foo =\long macro:
#1->bar.

```

If the inner command is defined with an optional argument, then `\@show@newcommand` is also used.

The value of `\escapechar` is deliberately not enforced, so `\ShowCommand` behaves more like `\show`.

```

566 \long\def\@show@DeclareRobustCommand#1{%
567   \typeout{> \string#1=robust macro:}%
568   \typeout{->\@expl@cs@replacement@spec@@N#1.^~J}%
569   \expandafter\show@kernel@robust@command
570   \csname\@expl@cs@to@str@@N#1 \endcsname}%
571 \long\def\show@kernel@robust@command#1{%
572   \robust@command@chk@safe#1%
573   {\@if@newcommand#1%
574     {\@show@newcommand}%
575     {\show}%
576     {\show}%
577   #1}

```

(End definition for `\@if@DeclareRobustCommand` and others.)

1.5.4 Commands defined with \newcommand (with optional argument)

\@if@newcommand A command \cmd (or \cmd_ if it was defined with \DeclareRobustCommand) with an optional argument will expand to \protected@testopt\cmd{\cmd{<opt>}}. To check that we look at the first three tokens in the expansion of \cmd, and return true or false accordingly.

This test *requires* that the command be a parameterless macro, otherwise it will not work (and probably break). This is ensured with \robust@command@chk@safe before calling \@if@newcommand.

```

578 \long\def\@if@newcommand#1{%
579   \edef\reserved@a{%
580     \noexpand\@protected@testopt
581     \noexpand#1%
582     \expandafter\noexpand\csname\@backslashchar\expl@cs@to@str@@N#1\endcsname}%
583   \edef\reserved@b{%
584     \unexpanded\expandafter\expandafter\expandafter
585       {\expandafter\@car\@cube#1{}{}{\@nil}}}%
586   \ifx\reserved@a\reserved@b
587     \expandafter\@firstoftwo
588   \else
589     \expandafter\@secondoftwo
590   \fi}

```

Then, if a command \cmd takes an optional argument, we copy it to \foo by defining the latter to expand to \protected@testopt\foo\foo{<opt>}.

```

591 \long\def\@copy@newcommand#1#2{%
592   \edef#1{\noexpand\@protected@testopt
593     \noexpand#1%
594     \expandafter\noexpand\csname\@backslashchar\expl@cs@to@str@@N#1\endcsname
595     \unexpanded\expandafter\expandafter\expandafter
596       {\expandafter\@gobblethree#2}}%
597   \expandafter
598   \let\csname\@backslashchar\expl@cs@to@str@@N#1\expandafter\endcsname
599   \csname\@backslashchar\expl@cs@to@str@@N#2\endcsname}

```

A command being \shown here is guaranteed to have an optional argument. Start by showing the top-level expansion of the command (using \typeout to avoid TeX asking for interaction and extra context lines), then call \@show@newcommand@aux with the internal command, which contains the actual definition, and with the expansion of the command to extract the default value of the optional argument.

```

600 \long\def\@show@newcommand#1{%
601   \typeout{> \string#1=robust macro:}%
602   \typeout{->\@expl@cs@replacement@spec@@N#1.^~J}%
603   \expandafter\@show@newcommand@aux
604   \csname\@backslashchar\expl@cs@to@str@@N#1\expandafter\endcsname
605   \expandafter{#1}}

```

For a macro defined with, say, \newcommand\foo[1][opt]{bar}, it will print:

```

> \foo=robust macro:
->\@protected@testopt \foo \\\foo {opt}.

> \\\foo=\long macro:

```

```
> default #1=opt.  
[#1]->bar.
```

If the command was defined with `\DeclareRobustCommand`, then another pair of lines show the top-level expansion `\protect\foo`.

The extra gymnastics with `\showtokens` ensures that `\showtokens` itself, and the internals of this macro aren't showed in the context lines.

```
606 \long\def\@show@newcommand@aux#1#2{%
607   \typeout{> \string#1=\@expl@cs@prefix@spec@@N#1macro:}%
608   \edef\reserved@a{%
609     default \string##1=\expandafter\detokenize@\gobblethree#2.^~J%
610     \@expl@cs@argument@spec@@N#1->\@expl@cs@replacement@spec@@N#1}%
611   \showtokens\expandafter\expandafter\expandafter{\expandafter\reserved@a}}
```

Now the rollback code.

```
612 </2ekernel | latexrelease>
613 <latexrelease>\EndIncludeInRelease
614 <latexrelease>\IncludeInRelease{0000-00-00}{\@if@DeclareRobustCommand}
615 <latexrelease>  {Undefine \@if@DeclareRobustCommand, \@if@newcommand,
616 <latexrelease>          \@copy@DeclareRobustCommand, \@copy@newcommand,
617 <latexrelease>          \@show@DeclareRobustCommand, \@show@newcommand}%
618 <latexrelease>\let\@if@DeclareRobustCommand\@undefined
619 <latexrelease>\let\@copy@DeclareRobustCommand\@undefined
620 <latexrelease>\let\@show@DeclareRobustCommand\@undefined
621 <latexrelease>\let\@if@newcommand\@undefined
622 <latexrelease>\let\@copy@newcommand\@undefined
623 <latexrelease>\let\@show@newcommand\@undefined
624 %
625 <latexrelease>\let\copy@kernel@robust@command\@undefined
626 <latexrelease>\let\show@kernel@robust@command\@undefined
627 <latexrelease>\let\@show@newcommand@aux\@undefined
628 <latexrelease>\EndIncludeInRelease
629 <*2ekernel>
```

(End definition for `\@if@newcommand` and others.)

1.6 Internal defining commands

These commands are used internally to define other L^AT_EX commands.

`\@ifundefined` Check if first arg is undefined or `\relax` and execute second or third arg depending,

```
630 </2ekernel>
631 <latexrelease>\IncludeInRelease{2018-04-01}{\@ifundefined}
632 <latexrelease>{Leave commands undefined in \@ifundefined}%
633 <*2ekernel | latexrelease>
```

Version using `\ifcsname` to avoid defining undefined tokens to `\relax`. Defined here to simplify using unmatched `\fi`.

```
634 \def\@ifundefined#1{%
635   \ifcsname#1\endcsname\@ifundefined@#1\else\@ifundefined@#1\fi{#1}}
636 \long\def\@ifundefined@#1\fi{#2}{\fi
637   \expandafter\ifx\csname #2\endcsname\relax
638   \quad\@ifundefined@#1\fi
639   \quad\@secondoftwo}
```

```
641 \long\def\@ifundefined@d@i\fi#1#2#3{\fi #2}
```

Now test of engine.

```
642 \ifx\numexpr\@undefined
```

Classic version (should not be needed as etex is assumed).

```
643 \def\@ifundefined#1{%
644   \expandafter\ifx\csname#1\endcsname\relax
645     \expandafter\@firstoftwo
646   \else
647     \expandafter\@secondoftwo
648   \fi}
649 \else\ifx\directlua\@undefined
```

Use the \ifcsname defined above.

```
650 \else
```

Optimised version for LuaTeX, using \lastnamedcs

```
651 \def\@ifundefined#1{%
652   \ifcsname#1\endcsname
653     \expandafter\ifx\lastnamedcs\relax\else\@ifundefined@d@i\fi
654   \fi
655   \@firstoftwo}
656 \long\def\@ifundefined@d@i#1#2#3#4#5{#1#2#5}
657 \fi
658 \fi
659 {/2ekernel | latexrelease}
660 {/latexrelease}\EndIncludeInRelease
661 {/latexrelease}\IncludeInRelease{0000-00-00}{\@ifundefined}
662 {/latexrelease}{Leave commands undefined in \@ifundefined}%
663 {/latexrelease}\def\@ifundefined#1{%
664 {/latexrelease} \expandafter\ifx\csname#1\endcsname\relax
665 {/latexrelease} \expandafter\@firstoftwo
666 {/latexrelease} \else
667 {/latexrelease} \expandafter\@secondoftwo
668 {/latexrelease} \fi}
669 {/latexrelease}\EndIncludeInRelease
670 {/2ekernel}
```

(End definition for \@ifundefined.)

\@qend The following define \@qend and \@qrelax to be the strings ‘end’ and ‘relax’ with the characters \catcodel 12.

```
671 \edef\@qend{\expandafter\@cdr\string\end\@nil}
672 \edef\@qrelax{\expandafter\@cdr\string\relax\@nil}
```

(End definition for \@qend and \@qrelax.)

\@ifnextchar \@ifnextchar peeks at the following character and compares it with its first argument. If both are the same it executes its second argument, otherwise its third.

```
673 \long\def\@ifnextchar#1#2#3{%
674   \let\reserved@d=#1%
675   \def\reserved@a{#2}%
676   \def\reserved@b{#3}%
677   \futurelet\@let@token\@ifnch}
```

(End definition for `\@ifnextchar`.)

- `\kernel@ifnextchar` This macro is the kernel version of `\@ifnextchar` which is used in a couple of places to prevent the AMS variant from being used since in some places this produced chaos (for example if an `fd` file is loaded in a random place then the optional argument to `\ProvidesFile` could get printed there instead of being written only in the log file. This happened when there was a space or a newline between the mandatory and optional arguments! It should really be fixed in the `amsmath` package one day, but...

Note that there may be other places in the kernel where this version should be used rather than the original, but variable, version.

```
678 \let\kernel@ifnextchar\@ifnextchar
```

(End definition for `\kernel@ifnextchar`.)

- `\@ifnch` `\@ifnch` is a tricky macro to skip any space tokens that may appear before the character in question. If it encounters a space token, it calls `xifnch`.

```
679 \def\@ifnch{%
680   \ifx\@let@token\@sptoken
681     \let\reserved@c\@xifnch
682   \else
683     \ifx\@let@token\reserved@d
684       \let\reserved@c\reserved@a
685     \else
686       \let\reserved@c\reserved@b
687     \fi
688   \fi
689 }
```

(End definition for `\@ifnch`.)

- `\@sptoken` The following code makes `\@sptoken` a space token. It is important here that the control sequence `\:` consists of a non-letter only, so that the following whitespace is significant. Together with the fact that the equal sign in a `\let` may be followed by only one optional space the desired effect is achieved. NOTE: the following hacking must precede the definition of `\:` as math medium space.

```
690 \def\:{\let\@sptoken= } \: % this makes \@sptoken a space token
```

(End definition for `\@sptoken`.)

- `\@xifnch` In the following definition of `\@xifnch`, `\:` is again used to get a space token as delimiter into the definition.

```
691 \def\:{\@xifnch} \expandafter\def\:{\futurelet\@let@token\@ifnch}
```

(End definition for `\@xifnch`.)

- `\@ifstar` The new implementation below avoids passing the *true code* through one more `\def` than the *false code*, which previously meant that `#` had to be written as `####` in one argument, but `##` in the other. The `*` is gobbled by `\@firstoftwo`.

```
692 \def\@ifstar#1{\@ifnextchar *{\@firstoftwo{#1}}}
```

(End definition for `\@ifstar`.)

```

\@dblarg
\@xdblarg 693 \long\def\@dblarg#1{\kernel@ifnextchar[{\#1}{\@xdblarg{#1}}]}
694 \long\def\@xdblarg#1#2[#1[{\#2}]{#2}]

(End definition for \@dblarg and \@xdblarg.)

```

- \@sanitize** The command `\@sanitize` changes the catcode of all special characters except for braces to ‘other’. It can be used for commands like `\index` that want to write their arguments verbatim. Needless to say, this command should only be executed within a group, or chaos will ensue.

```

695 \def\@sanitize{\@makeother\ \@makeother\\ \@makeother\$ \@makeother\&%
696 \@makeother\#\@makeother\^ \@makeother\_ \@makeother\% \@makeother\~}

```

(End definition for \@sanitize.)

- \@onelvel@sanitize** This makes the whole “meaning” of #1 (its one-level expansion) into catcode 12 tokens: it could be used in `\DeclareRobustCommand`.

If it is to be used on default float specifiers, this should be done when they are defined.

```

697 \def \@onelvel@sanitize #1{%
698   \edef #1{\expandafter\strip@prefix
699     \meaning #1}%
700 }

```

(End definition for \@onelvel@sanitize.)

- \string@makeletter** Iterates through a string, turning each alphabetic character into a catcode-11 token (partly undoes a `\detokenize`). Useful for `\ifx`-based string comparisons where `\detokenize`-ing the other string would break too much code.

The macro uses `expl3`’s `\Expl@str@map@function@@NN` to iterate on the string (without losing spaces) and applies `\string@makeletter` on each character. The latter checks if character is between a–z or A–Z, and uses `\@alph` or `\@Alph` to get the corresponding catcode-11 token. Other tokens are passed through unchanged.

```

701 </2ekernel>
702 <latexrelease>\IncludeInRelease{2020/10/01}{\string@makeletter}
703 <latexrelease> {Add \string@makeletter}%
704 <2ekernel | latexrelease>
705 \def\string@makeletter#1{%
706   \Expl@str@map@function@@NN#1\string@makeletter}
707 \def\@string@makeletter#1{%
708   \char@if@alph{#1}%
709   {\Expl@char@generate@@nn{'#1}{11}}%
710   {'#1}}
711 \def\char@if@alph#1{%
712   \ifnum0\ifnum`#1<'A 1\fi\ifnum`#1>'z 1\fi
713   \if\ifnum`#1>'Z @\fi\ifnum`#1<'a @\fi01\fi>0
714   \expandafter\@secondoftwo
715 \else
716   \expandafter\@firstoftwo
717 \fi}
718 </2ekernel | latexrelease>
719 <latexrelease>\EndIncludeInRelease
720 %

```

```

721 〈\latexrelease〉\IncludeInRelease{0000/00/00}{\string@makeletter}
722 〈\latexrelease〉 {Undefined \string@makeletter}%
723 〈\latexrelease〉\let\string@makeletter\@undefined
724 〈\latexrelease〉\let\@string@makeletter\@undefined
725 〈\latexrelease〉\let\char@if@alph\@undefined
726 〈\latexrelease〉\EndIncludeInRelease
727 {*ekernel}

```

(*End definition for \string@makeletter, \@string@makeletter, and \char@if@alph.*)

\makeatletter Make internal control sequences accessible or inaccessible.

\makeatother 728 \DeclareRobustCommand\makeatletter{\catcode`@11\relax}
729 \DeclareRobustCommand\makeatother{\catcode`@12\relax}

(*End definition for \makeatletter and \makeatother.*)

2 Discretionary Hyphenation

\-

\@dischypf

Moved here to be after the definition of \DeclareRobustCommand.

The primitive \- command adds a discretionary hyphen using the current font's \hyphenchar. Monospace fonts are usually declared with \hyphenchar set to -1 to suppress hyphenation.

\ATEX, from \ATEX2.09 in 1986 defined \- by

```
\def\-\{\discretionary{-}{ }{ }\}
```

The following comment was added when these commands were first set up, 19 April 1986:

the \- command is redefined to allow it to work in the \ttfamily type style, where automatic hyphenation is suppressed by setting \hyphenchar to -1. The original primitive \TeX definition is saved as \@@hyph just in case anyone needs it.

\ATEX 2 ε , between 1993 and 2017, had a comment at this point saying that the definition "would probably change" because the definition always uses -. The definition used below was given in comments at this point during time.

In 2017 we finally enabled this definition by default, with the older \ATEX definition accessible via \textrm{latextext} as usual.

In \Lua\ATEX the primitive definition of \- is used directly because it's use of extended hyphenation parameters means that \- works correctly even with \hyphenchar set to -1. This change makes \- under \Lua\ATEX compatible with language specific hyphenation characters.

Temporary definition of \@latextext, final definition is later.

```

730 \def\@latextext#1{%
731   \ifx\directlua\undefined
732     \IncludeInRelease{2020/10/01}{\-}{Use primitive \- in \Lua\LaTeX}%
733   {*ekernel | \textrm{latextext}}%
734   \ifx\directlua\undefined
735     \DeclareRobustCommand{\-}{%
736       \discretionary{}{ }{ }%
737       \char \ifnum\hyphenchar<\z@

```

```

738           \defaulthyphenchar
739     \else
740       \hyphenchar\font
741     \fi
742   }{}{}%
743 }
744 \else
745   \let\-\@hyph
746 \fi
747 </2ekernel | latexrelease>
748 <latexrelease>\EndIncludeInRelease
749 <latexrelease>\IncludeInRelease{2017/04/15}{\Use \hyphenchar in \-}%
750 <latexrelease>\DeclareRobustCommand{\-}{%
751 <latexrelease> \discretionary{%
752 <latexrelease>   \char \ifnum\hyphenchar\font<\z@
753 <latexrelease>           \defaulthyphenchar
754 <latexrelease>           \else
755 <latexrelease>             \hyphenchar\font
756 <latexrelease>           \fi
757 <latexrelease>         }{}{}%
758 <latexrelease>}
759 <latexrelease>\EndIncludeInRelease
760 <latexrelease>\IncludeInRelease{0000/00/00}{\Use \hyphenchar in \-}%
761 <latexrelease>\def\-\{\discretionary{-}{}{}}
762 <latexrelease>\EndIncludeInRelease

763 <*2ekernel | latexrelease>
764 \let\@dischyp=\
765 </2ekernel | latexrelease>
766 <*2ekernel>

(End definition for \- and \@dischyp.)
Delayed from ltvers.dtx
767 \newif\if@includeinrelease
768 \Q@includeinreleasefalse
Delayed from ltplain.dtx
769 </2ekernel>
770 <*2ekernel | latexrelease>
771 <latexrelease>\IncludeInRelease{2019/10/01}%
772 <latexrelease>           {\allowbreak}{\Make various commands robust}%
773 \MakeRobust\allowbreak
774 \MakeRobust\bigbreak
775 \MakeRobust\break
776 \MakeRobust\dotfill
777 \MakeRobust\frenchspacing
778 \MakeRobust\goodbreak
779 \MakeRobust\hrulefill
780 \MakeRobust\medbreak
781 \MakeRobust\nobreak
782 \MakeRobust\nonfrenchspacing
783 \MakeRobust\obeyslines
784 \MakeRobust\obeyspaces
785 \MakeRobust\slash
786 \MakeRobust\smallbreak

```

```

787 \MakeRobust\strut
788 \MakeRobust\underbar
789 </2ekernel | latexrelease>
790 <latexrelease>\EndIncludeInRelease
791 <latexrelease>\IncludeInRelease{0000/00/00}%
792 <latexrelease> {\allowbreak}{\Make various commands robust}%
793 <latexrelease>
794 <latexrelease>\kernel@make@fragile\allowbreak
795 <latexrelease>\kernel@make@fragile\bigbreak
796 <latexrelease>\kernel@make@fragile\break
797 <latexrelease>\kernel@make@fragile\dotfill
798 <latexrelease>\kernel@make@fragile\frenchspacing
799 <latexrelease>\kernel@make@fragile\goodbreak
800 <latexrelease>\kernel@make@fragile\hrulefill
801 <latexrelease>\kernel@make@fragile\medbreak
802 <latexrelease>\kernel@make@fragile\nobreak
803 <latexrelease>\kernel@make@fragile\nonfrenchspacing
804 <latexrelease>\kernel@make@fragile\obeylines
805 <latexrelease>\kernel@make@fragile\obeyspaces
806 <latexrelease>\kernel@make@fragile\slash
807 <latexrelease>\kernel@make@fragile\smallbreak
808 <latexrelease>\kernel@make@fragile\strut
809 <latexrelease>\kernel@make@fragile\underbar
810 <latexrelease>
811 <latexrelease>\EndIncludeInRelease
812 <*2ekernel>

```

`\g@addto@macro` Globally add to the end of a macro. This macro is used by the kernel to add to its internal hooks.

```

813 \long\def\g@addto@macro#1#2{%
814   \begingroup
815     \toks@\expandafter{\#1#2}%
816     \xdef#1{\the\toks@}%
817   \endgroup}

```

(*End definition for \g@addto@macro.*)

```
818 </2ekernel>
```

File g

ltcmd.dtx

1 Creating document commands

Document commands should be created using the tools provided by this module: `\NewDocumentCommand`, etc., in almost all cases. This allows clean separation of document-level syntax from code-level interfaces. Users have a need to create new document commands, and as such a significant amount of documentation for `ltcmd` is provided as part of `usrguide3`. Here, additional material aimed at programmers is provided

```
1 (@@=cmd)
2 {*2ekernel}
3 \message{document commands,}
4 {/2ekernel}
```

`ltcmd` code contains an `\^@` character, which usually has catcode 15, so `\IncludeInRelease` will break when this code is being skipped, so we'll save the catcode of `\^@` to restore later:

```
5 {*2ekernel | latexrelease}
6 {latexrelease}\edef\@latexrelease@catcode@null{\the\catcode'\^@ }
7 {latexrelease}\catcode'\^@=12
8 \ExplSyntaxOn
9 {latexrelease}\NewModuleRelease{2020/10/01}{ltcmd}
10 {latexrelease} {Document~command~parser} %
```

1.1 Variables and constants

`\l__cmd_arg_spec_tl` Holds the argument specification after normalization of shorthands.

```
11 \tl_new:N \l__cmd_arg_spec_tl
```

`\l__cmd_args_tl` Token list variable for grabbed arguments.

```
12 \tl_new:N \l__cmd_args_tl
```

`\l__cmd_args_i_tl` Hold the modified arguments when dealing with default values or processors.

`\l__cmd_args_ii_tl`

```
13 \tl_new:N \l__cmd_args_i_tl
14 \tl_new:N \l__cmd_args_ii_tl
```

`\l__cmd_current_arg_int` The number of the current argument being set up: this is used to make sure there are at most 9 arguments, then for creating the expandable auxiliary functions and knowing how many arguments the code function should take.

```
15 \int_new:N \l__cmd_current_arg_int
```

\l__cmd_defaults_bool The boolean indicates whether there are any argument with default value other than -*NoValue*-; the token list holds the code to determine these default values in terms of other arguments.

```
16 \bool_new:N \l__cmd_defaults_bool  
17 \tl_new:N \l__cmd_defaults_tl
```

\l__cmd_environment_bool Generating environments uses the same mechanism as generating functions. However, full processing of arguments is always needed for environments, and so the function-generating code needs to know this. This variable is also used at run time to give correct error messages.

```
18 \bool_new:N \l__cmd_environment_bool
```

\l__cmd_environment_str Name of the environment, used at definition time and at run time.

```
19 \str_new:N \l__cmd_environment_str
```

\l__cmd_expandable_bool Used to indicate if an expandable command is being generated, as this affects both the acceptable argument types and how they are implemented.

```
20 \bool_new:N \l__cmd_expandable_bool
```

\l__cmd_expandable_aux_name_tl

Used to create pretty-printing names for the auxiliaries: although the immediate definition does not vary, the full expansion does and so it does not count as a constant.

```
21 \tl_new:N \l__cmd_expandable_aux_name_tl  
22 \tl_set:Nn \l__cmd_expandable_aux_name_tl  
23 {  
24     \l__cmd_function_tl \c_space_tl  
25     ( arg~ \int_use:N \l__cmd_current_arg_int )  
26 }
```

\g__cmd_grabber_int Used (in exceptional cases) to get unique names for grabbers used by expandable commands.

```
27 \int_new:N \g__cmd_grabber_int
```

\l__cmd_fn_tl For passing the pre-formed name of the auxiliary to be used as the parsing function.

```
28 \tl_new:N \l__cmd_fn_tl
```

\l__cmd_fn_code_tl For passing the pre-formed name of the auxiliary that contains the actual code.

```
29 \tl_new:N \l__cmd_fn_code_tl
```

\l__cmd_function_tl Holds the control sequence name of the function currently being defined: used to avoid passing this as an argument and to avoid repeated use of \cs_to_str:N.

³⁰ \tl_new:N \l__cmd_function_tl

\l__cmd_grab_expandably_bool

When defining a non-expandable command, indicates whether the arguments can all safely be grabbed by expandable grabbers. This is to support abuses of *xparse* that use protected functions inside csname constructions.

³¹ \bool_new:N \l__cmd_grab_expandably_bool

\l__cmd_obey_spaces_bool For trailing optionals.

³² \bool_new:N \l__cmd_obey_spaces_bool

\l__cmd_last_delimiters_tl Holds the delimiters (first tokens) of all optional arguments since the previous mandatory argument, to warn about cases where it would be impossible to omit optional arguments completely because the following mandatory argument has the same delimiter as one of the optional arguments.

³³ \tl_new:N \l__cmd_last_delimiters_tl

\l__cmd_long_bool Used to indicate that an argument is long, on a per-argument basis.

³⁴ \bool_new:N \l__cmd_long_bool

\l__cmd_m_args_int The number of *m* arguments: if this is the same as the total number of arguments, then a short-cut can be taken in the creation of the grabber code.

³⁵ \int_new:N \l__cmd_m_args_int

\l__cmd_prefixed_bool When preparing the signature of non-expandable commands, indicates that the current argument is affected by a processor or by + (namely is long).

³⁶ \bool_new:N \l__cmd_prefixed_bool

\l__cmd_process_all_tl
\l__cmd_process_one_tl
\l__cmd_process_some_bool When preparing the signature, the processors that will be applied to a given argument are collected in \l__cmd_process_one_tl, while \l__cmd_process_all_tl contains processors for all arguments. The boolean indicates whether there are any processors (to bypass the whole endeavour otherwise).

³⁷ \tl_new:N \l__cmd_process_all_tl

³⁸ \tl_new:N \l__cmd_process_one_tl

³⁹ \bool_new:N \l__cmd_process_some_bool

`\l__cmd_saved_args_tl` Stores `\l__cmd_args_tl` to deal with space-trimming of b-type arguments.

40 `\tl_new:N \l__cmd_saved_args_tl`

`\l__cmd_signature_tl` Used when constructing the signature (code for argument grabbing) to hold what will become the implementation of the main function. When arguments are grabbed (at point of use of the command/environment), it also stores the code for grabbing the remaining arguments.

41 `\tl_new:N \l__cmd_signature_tl`

`\l__cmd_some_obey_spaces_bool`
`\l__cmd_some_long_bool`
`\l__cmd_some_short_bool`

These flags are set while normalizing the argument specification. The `obey_spaces` one is used to detect when ! is used on an argument that is not a trailing optional argument. The other two are used to check whether all short arguments appear before long arguments: this is needed to grab arguments expandably. As soon as the first long argument is seen (other than t-type, whose long status is ignored) the `some_long` flag is set. The `some_short` flag is used for expandable commands, to know whether to define a short auxiliary too.

42 `\bool_new:N \l__cmd_some_obey_spaces_bool`
43 `\bool_new:N \l__cmd_some_long_bool`
44 `\bool_new:N \l__cmd_some_short_bool`

`\l__cmd_tmp_prop`
`\l__cmd_tmptl`
`\l__cmd_tmpb_tl`

Scratch space.
45 `\prop_new:N \l__cmd_tmp_prop`
46 `\tl_new:N \l__cmd_tmptl`
47 `\tl_new:N \l__cmd_tmpb_tl`
48 `\cs_new_eq:NN \l__cmd_tmp:w ?`

(End definition for `\l__cmd_tmp:w`.)

With `xparse`, information about commands being (re)defined was switched off by default, unless the `log-declarations` package option was used, so here we'll switch that off as well.

49 `\msg_redirect_module:nnn { cmd } { info } { none }`

Also add `cmd` to the `LaTeX` messages.

50 `\prop_gput:Nnn \g_msg_module_type_prop { cmd } { LaTeX }`

1.2 Declaring commands and environments

The main functions for creating commands set the appropriate flag then use the same internal code to do the definition.

51 `\cs_new_protected:Npn \l__cmd_declare_cmd:Nnn`
52 {
53 `\bool_set_false:N \l__cmd_expandable_bool`

```

54      \__cmd_declare_cmd_aux:Nnn
55  }
56 \cs_new_protected:Npn \__cmd_declare_expandable_cmd:Nnn
57 {
58     \bool_set_true:N \l__cmd_expandable_bool
59     \__cmd_declare_cmd_aux:Nnn
60 }

```

The first stage is to log information, both for the user in the log and for programmatic use in a property list of all declared commands.

```

61 \cs_new_protected:Npn \__cmd_declare_cmd_aux:Nnn #1#2#3
62 {
63     \cs_if_exist:NTF #1
64     {
65         \msg_info:nxxx { cmd } { redefine }
66         { \token_to_str:N #1 } { \tl_to_str:n {#2} }
67     }
68     {
69         \bool_lazy_or:nnT
70         { \cs_if_exist_p:c { \cs_to_str:N #1 ~ code } }
71         { \cs_if_exist_p:c { \cs_to_str:N #1 ~ defaults } }
72         {
73             \msg_warning:nnx { cmd } { unsupported-let }
74             { \token_to_str:N #1 }
75         }
76         \msg_info:nxxx { cmd } { define-command }
77         { \token_to_str:N #1 } { \tl_to_str:n {#2} }
78     }
79     \bool_set_false:N \l__cmd_environment_bool
80     \__cmd_declare_cmd_internal:Nnnn #1 {#2} {#3} { }
81 }

```

At definition time, the variable `\l__cmd_fn_tl` is only used for error messages. The real business of defining a document command starts with setting up the appropriate name, then normalizing the argument specification to get rid of shorthands.

```

82 \cs_new_protected:Npn \__cmd_declare_cmd_internal:Nnnn #1#2#3#4
83 {
84     \tl_set:Nx \l__cmd_function_tl { \cs_to_str:N #1 }
85     \tl_set:Nx \l__cmd_fn_tl
86     { \exp_not:c { \l__cmd_function_tl \c_space_tl } }
87     \__cmd_normalize_arg_spec:n {#2}
88     \exp_args:No \__cmd_prepare_signature:n \l__cmd_arg_spec_tl
89     \__cmd_declare_cmd_code:Nnn #1 {#2} {#3}
90     #4
91     \__cmd_break_point:n {#2}
92 }

```

(End definition for `__cmd_declare_cmd:Nnn` and others.)

`__cmd_break_point:n` A marker used to escape from creating a definition if necessary.

```

93 \cs_new_eq:NN \__cmd_break_point:n \use_none:n

```

(End definition for `__cmd_break_point:n`.)

```
\_\_cmd\_declare\_cmd\_code:Nnn
\_\_cmd\_declare\_cmd\_code\_aux:Nnn
\_\_cmd\_declare\_cmd\_code\_expandable:Nnn
```

The appropriate auxiliary is called.

```
94 \cs_new_protected:Npn \_\_cmd\_declare\_cmd\_code:Nnn
95 {
96     \bool_if:NTF \l__cmd_grab_expandably_bool
97     { \_\_cmd_declare_cmd_code_expandable:Nnn }
98     { \_\_cmd_declare_cmd_code_aux:Nnn }
99 }
```

Standard functions call `__cmd_start:nNNnnn`, which receives the argument specification, an auxiliary used for grabbing arguments, an auxiliary containing the code, and then the signature, default arguments, and processors.

```
100 \cs_new_protected:Npn \_\_cmd_declare_cmd_code_aux:Nnn #1#2#3
101 {
102     \cs_generate_from_arg_count:cNnn
103     { \l__cmd_function_tl \c_space_tl code }
104     \cs_set_protected:Npn \l__cmd_current_arg_int {#3}
105     \cs_set_protected_nopar:Npx #1
106     {
107         \bool_if:NTF \l__cmd_environment_bool
108         {
109             \_\_cmd_start_env:nnnnn { \exp_not:n {#2} }
110             { \l__cmd_environment_str }
111         }
112         {
113             \_\_cmd_start:nNNnnn { \exp_not:n {#2} }
114             \exp_not:c { \l__cmd_function_tl \c_space_tl }
115             \exp_not:c { \l__cmd_function_tl \c_space_tl code }
116         }
117         { \exp_not:o \l__cmd_signature_tl }
118         {
119             \bool_if:NT \l__cmd_defaults_bool
120             { \exp_not:o \l__cmd_defaults_tl }
121         }
122         {
123             \bool_if:NT \l__cmd_process_some_bool
124             { \exp_not:o \l__cmd_process_all_tl }
125         }
126     }
127 }
```

Expandable functions and functions whose arguments can be grabbed expandably call `__cmd_start_expandable:nNNNNn`, which receives the argument specification, four auxiliaries (two for grabbing arguments, one for the code, and one for default arguments), and finally the signature. Non-expandable functions that take this branch should nevertheless be protected, as well as their `code` function. They will only be expanded in contexts such as constructing a `cname`. The two grabbers (named after the function with one or two spaces) are needed when there are both short and long arguments; otherwise the same grabber is included twice in the definition. If all arguments are long or all are short (only) grabber is defined correspondingly to be long/short. Otherwise two grabbers are defined, one long, one short.

```
128 \cs_new_protected:Npn \_\_cmd_declare_cmd_code_expandable:Nnn #1#2#3
129 {
130     \exp_args:Ncc \cs_generate_from_arg_count:NNnn
131     { \l__cmd_function_tl \c_space_tl code }
```

```

132   { cs_set \bool_if:NF \l__cmd_expandable_bool { _protected } :Npn }
133   \l__cmd_current_arg_int {#3}
134 \bool_if:NT \l__cmd_defaults_bool
135 {
136   \use:x
137   {
138     \cs_generate_from_arg_count:cNnn
139     { \l__cmd_function_tl \c_space_tl defaults }
140     \cs_set:Npn \l__cmd_current_arg_int
141     { \exp_not:o \l__cmd_defaults_tl }
142   }
143 }
144 \bool_if:NTF \l__cmd_expandable_bool
145 { \cs_set_nopar:Npx } { \cs_set_protected_nopar:Npx } #1
146 {
147   \exp_not:N \l__cmd_start_expandable:nNNNNn
148   { \exp_not:n {#2} }
149   \exp_not:c { \l__cmd_function_tl \c_space_tl }
150   \exp_not:c
151   {
152     \l__cmd_function_tl \c_space_tl
153     \bool_if:NT \l__cmd_some_short_bool
154     { \bool_if:NT \l__cmd_some_long_bool { \c_space_tl } }
155   }
156   \exp_not:c { \l__cmd_function_tl \c_space_tl code }
157   \bool_if:NTF \l__cmd_defaults_bool
158   { \exp_not:c { \l__cmd_function_tl \c_space_tl defaults } }
159   { ? }
160   { \exp_not:o \l__cmd_signature_tl }
161 }
162 \bool_if:NTF \l__cmd_some_long_bool
163 {
164   \bool_if:NT \l__cmd_some_short_bool
165   {
166     \cs_set_nopar:cpx { \l__cmd_function_tl \c_space_tl \c_space_tl }
167     ##1##2 { ##1 {##2} }
168   }
169   \cs_set:cpx
170 }
171 { \cs_set_nopar:cpx }
172 { \l__cmd_function_tl \c_space_tl } ##1##2 { ##1 {##2} }
173 }

```

(End definition for `\l__cmd_declare_cmd_code:Nnn`, `\l__cmd_declare_cmd_code_aux:Nnn`, and
`\l__cmd_declare_cmd_code_expandable:Nnn`.)

`\l__cmd_declare_env:nnnn`
`\l__cmd_declare_env_internal:nnnn`

The lead-off to creating an environment is much the same as that for creating a command: issue the appropriate message, store the argument specification then hand off to an internal function.

```

174 \cs_new_protected:Npn \l__cmd_declare_env:nnnn #1#2
175 {
176   \str_set:Nx \l__cmd_environment_str {#1}
177   \str_set:Nx \l__cmd_environment_str
178   { \tl_trim_spaces:o { \l__cmd_environment_str } }

```

```

179   \cs_if_exist:cTF { \l__cmd_environment_str }
180   {
181     \msg_info:nxxx { cmd } { redefine-env }
182     { \l__cmd_environment_str } { \tl_to_str:n {#2} }
183   }
184   {
185     \msg_info:nxxx { cmd } { define-env }
186     { \l__cmd_environment_str } { \tl_to_str:n {#2} }
187   }
188 \bool_set_false:N \l__cmd_expandable_bool
189 \bool_set_true:N \l__cmd_environment_bool
190 \exp_args:NV \__cmd_declare_env_internal:nnnn
191   \l__cmd_environment_str {#2}
192 }

```

Creating a document environment requires a few more steps than creating a single command. In order to pass the arguments of the command to the end of the function, it is necessary to store the grabbed arguments. To do that, the function used at the end of the environment has to be redefined to contain the appropriate information. To minimize the amount of expansion at point of use, the code here is expanded now as well as when used. The last argument of `__cmd_declare_cmd_internal:Nnnn` is only run if the definition succeeded. In package mode this ensures that the original definition of the environment is not changed if the definition fails for any reason. This also avoids an error when defining the `end_aux` function when the user asks for more than 9 arguments.

```

193 \cs_new_protected:Npn \__cmd_declare_env_internal:nnnn #1#2#3#4
194   {
195     \exp_args:Nc \__cmd_declare_cmd_internal:Nnnn { environment~ #1 } {#2}
196     {#3}
197   {
198     \cs_set_nopar:cpx { environment~ #1 ~end~ }
199     { \exp_not:c { environment~ #1 ~end-aux~ } }
200     \cs_generate_from_arg_count:cNnn
201     { environment~ #1 ~end-aux~ } \cs_set:Npn
202       \l__cmd_current_arg_int {#4}
203     \cs_set_eq:cc {#1} { environment~ #1 }
204     \cs_set_eq:cc { end #1 } { environment~ #1 ~end~ }
205   }
206 }
207 \cs_new_protected:Npn \__cmd_set_environment_end:n #1
208   {
209     \cs_set_nopar:cpx { environment~ #1 ~end-aux~ }
210     {
211       \exp_not:c { environment~ #1 ~end-aux~ }
212       \exp_not:o \l__cmd_args_tl
213     }
214 }

```

(End definition for `__cmd_declare_env:nnnn` and `__cmd_declare_env_internal:nnnn`.)

1.3 Structure of `xparse` commands

`__cmd_start_env:nnnn`
`__cmd_start:nNNnn`

For error messages that occur during run-time when getting arguments of environments it is necessary to keep track of the environment name. We begin non-expandable commands with a token equal to `\scan_stop:`, whose name gives a reasonable error message

if the command is used inside a csname and protects against f-expansion. This is useless for environments since \begin is already not expandable. Both the command and environment codes start with \group_align_safe_begin:, then \cmd_run_code: (used by both) does \group_align_safe_end:, so that delimited arguments may be grabbed in alignments if they contain and alignment tab token (see latex3/latex3/issues/839).

```

215 \cs_new_protected:Npn \__cmd_start_env:nnnnn #1#2
216 {
217     \conditionally@traceoff
218     \group_align_safe_begin:
219     \str_set:Nn \l__cmd_environment_str {#2}
220     \bool_set_true:N \l__cmd_environment_bool
221     \__cmd_start_aux:ccnnnn
222     { environment~\l__cmd_environment_str \c_space_tl }
223     { environment~\l__cmd_environment_str \c_space_tl code }
224     {#1}
225 }
226 \cs_new_protected:Npx \__cmd_start:nNnnn #1#2#3
227 {
228     \exp_not:c { xparse~function~is~not~expandable }
229     \exp_not:N \conditionally@traceoff
230     \exp_not:N \group_align_safe_begin:
231     \exp_not:n { \bool_set_false:N \l__cmd_environment_bool }
232     \exp_not:N \__cmd_start_aux:NNnnn
233     #2 #3 {#1}
234 }
```

(End definition for __cmd_start_env:nnnnn and __cmd_start:nNnnn.)

__cmd_start_aux:NNnnn
__cmd_start_aux:ccnnnn

This sets up a few variables to minimize the boilerplate code included in all xparse-defined commands. It then runs the grabbers #4. Again, the argument specification #1 is only for diagnostics.

```

235 \cs_new_protected:Npn \__cmd_start_aux:NNnnnn #1#2#3#4#5#6
236 {
237     \tl_clear:N \l__cmd_args_tl
238     \tl_set:Nn \l__cmd_fn_tl {#1}
239     \tl_set:Nn \l__cmd_fn_code_tl {#2}
240     \tl_set:Nn \l__cmd_defaults_tl {#5}
241     \tl_set:Nn \l__cmd_process_all_tl {#6}
242     #4
243     \__cmd_run_code:
244 }
245 \cs_generate_variant:Nn \__cmd_start_aux:NNnnnn { cc }
```

(End definition for __cmd_start_aux:NNnnnn.)

__cmd_run_code:
After arguments are grabbed, this function is responsible for inserting default values, running processors, and finally doing \group_align_safe_end: as promised, and running the code.

```

246 \cs_new_protected:Npn \__cmd_run_code:
247 {
248     \tl_if_empty:NF \l__cmd_defaults_tl { \__cmd_defaults: }
249     \tl_if_empty:NF \l__cmd_process_all_tl { \__cmd_args_process: }
250     \bool_if:NT \l__cmd_environment_bool
251         { \exp_args:No \__cmd_set_environment_end:n \l__cmd_environment_str }
```

```

252     \group_align_safe_end:
253     \conditionally@traceon
254     \exp_after:wN \l__cmd_fn_code_tl \l__cmd_args_tl
255 }

```

(End definition for `__cmd_run_code`..)

`__cmd_defaults:` `__cmd_defaults_def:` `__cmd_defaults_def:nn` `__cmd_defaults_def:nnn` `__cmd_defaults_aux:` `__cmd_defaults_error:w`

First construct `__cmd_tmp:w` (see below) that will receive the arguments found so far and determine default values for any missing argument. Then call it repeatedly until the set of arguments stabilizes. Since that could lead to an infinite loop we only call it up to nine times, the maximal number needed for stabilization if there is a chain of arguments that depend on each other. If that fails to stabilize raise an error.

```

256 \cs_new_protected:Npn \__cmd_defaults:
257 {
258     \__cmd_defaults_def:
259     \tl_set_eq:NN \l__cmd_args_i_tl \l__cmd_args_tl
260     \__cmd_defaults_aux: \__cmd_defaults_aux: \__cmd_defaults_aux:
261     \__cmd_defaults_aux: \__cmd_defaults_aux: \__cmd_defaults_aux:
262     \__cmd_defaults_aux: \__cmd_defaults_aux: \__cmd_defaults_aux:
263     \__cmd_defaults_error:w
264     \q_recursion_stop
265     \tl_set_eq:NN \l__cmd_args_tl \l__cmd_args_i_tl
266 }
267 \cs_new_protected:Npn \__cmd_defaults_aux:
268 {
269     \tl_set:Nx \l__cmd_args_ii_tl
270     { \exp_after:wN \__cmd_tmp:w \l__cmd_args_i_tl }
271     \tl_if_eq:NNT \l__cmd_args_ii_tl \l__cmd_args_i_tl
272     { \use_none_delimit_by_q_recursion_stop:w }
273     \tl_set_eq:NN \l__cmd_args_i_tl \l__cmd_args_ii_tl
274 }
275 \cs_new_protected:Npn \__cmd_defaults_error:w \q_recursion_stop
276 {
277     \msg_error:nnx { cmd } { default-loop }
278     { \__cmd_environment_or_command: }
279 }

```

To construct `__cmd_tmp:w`, first go through the arguments found and the corresponding defaults, building a token list with `{#(arg number)}` for arguments found in the input (whose default will not be used) and otherwise `{\exp_not:n{(default)}}` for arguments whose default will be used.

```

280 \cs_new_protected:Npn \__cmd_defaults_def:
281 {
282     \tl_clear:N \l__cmd_tmpa_tl
283     \int_zero:N \l__cmd_current_arg_int
284     \__cmd_tl_mapthread_function:NNN \l__cmd_args_tl \l__cmd_defaults_tl
285     \__cmd_defaults_def:nn
286     \cs_generate_from_arg_count:NNVo \__cmd_tmp:w \cs_set:Npn
287     \l__cmd_current_arg_int \l__cmd_tmpa_tl
288 }
289 \cs_generate_variant:Nn \cs_generate_from_arg_count:NNnn { NNVo }
290 \cs_new_protected:Npn \__cmd_defaults_def:nn
291 {
292     \int_incr:N \l__cmd_current_arg_int

```

```

293      \exp_args:NV \__cmd_defaults_def:nnn \l__cmd_current_arg_int
294    }
295 \cs_new_protected:Npn \__cmd_defaults_def:nnn #1#2#3
296 {
297   \tl_put_right:Nx \l__cmd_tmpa_tl
298   {
299     {
300       \exp_not:N \exp_not:n
301       {
302         \tl_if_novalue:nTF {#2}
303         { \exp_not:o {#3} }
304         { \exp_not:n { ## #1 } }
305       }
306     }
307   }
308 }

```

(End definition for `__cmd_defaults:` and others.)

```

\__cmd_args_process:
\__cmd_args_process_loop:nn
\__cmd_args_process_aux:n

```

Loop through arguments (stored in `\l__cmd_args_tl`) and the corresponding processors (in `\l__cmd_process_all_tl`) simultaneously, apply all processors for each argument and store the result back into `\l__cmd_args_tl`. To allow processors to depend on other arguments, for every processor define a temporary auxiliary that receives all arguments `\l__cmd_args_tl`.

```

309 \cs_new_protected:Npn \__cmd_args_process:
310 {
311   \tl_clear:N \l__cmd_args_iitl
312   \__cmd_tl_mapthread_function:NNN
313   \l__cmd_args_tl
314   \l__cmd_process_all_tl
315   \__cmd_args_process_loop:nn
316   \tl_set_eq:NN \l__cmd_args_tl \l__cmd_args_iitl
317 }
318 \cs_new_protected:Npn \__cmd_args_process_loop:nn #1#2
319 {
320   \tl_set:Nn \ProcessedArgument {#1}
321   \tl_if_novalue:nF {#1}
322   { \tl_map_function:nN {#2} \__cmd_args_process_aux:n }
323   \tl_put_right:No \l__cmd_args_iitl
324   { \exp_after:wN { \ProcessedArgument } }
325 }
326 \cs_new_protected:Npn \__cmd_args_process_aux:n #
327 {
328   \cs_generate_from_arg_count:NNnn \__cmd_tmp:w \cs_set:Npn
329   { \tl_count:N \l__cmd_args_tl } {#1}
330   \exp_args:NNNo \exp_after:wN \__cmd_tmp:w \l__cmd_args_tl
331   { \ProcessedArgument }
332 }

```

(End definition for `__cmd_args_process:, __cmd_args_process_loop:nn, and __cmd_args_process_aux:n`.)

```
\__cmd_start_expandable:nNNNN
```

This is called for all expandable commands. #6 is the signature, responsible for grabbing arguments. #5 is used to determine default values (or is ? if there are none). #4 is

the code to run. #2 and #3 are functions (named after the command) that grab a single argument in the input stream (#3 is short). The argument specification #1 is only used by diagnostic functions. Same as for the non-expandable version, this starts with \group_align_safe_begin:, which expands to nothing, so may be safely used in an expandable context.

```

333 \cs_new:Npn \__cmd_start_expandable:nNNNNn #1#2#3#4#5#6
334 {
335     \group_align_safe_begin:
336     #6 \__cmd_end_expandable:NNw #5 #4 \q__cmd #2#3
337 }

```

(End definition for __cmd_start_expandable:nNNNNn.)

Followed by a function #1 to determine default values (or ? if there are no defaults), the code #2, arguments that have been grabbed, then \q__cmd and two generic grabbers. The idea to find default values is similar to the non-expandable case but we cannot define an auxiliary function, so at every step in the loop we need to go through all arguments searching for which ones started out as -NoValue- and replacing these by the newly computed values. In fact we need to keep track of three versions of all arguments: the original version, the previous version with default values, and the currently built version (first argument of __cmd_end_expandable_defaults:nnnNNn).

```

338 \cs_new:Npn \__cmd_end_expandable:NNw #1#2
339 {
340     \__cmd_end_expandable_aux:w #1#2 \prg_do_nothing:
341     { \exp_args:No \__cmd_end_expandable_aux:nNNNN {#3} #1 #2 }
342 \cs_new:Npn \__cmd_end_expandable_aux:nNNNN #1#2#3#4#5
343 {
344     \token_if_eq_charcode:NNT ? #2 { \exp_after:wN \use_iv:nnnn }
345     \__cmd_end_expandable_defaults:nnnNNn {#1} { } {#1} #2#3
346     { } { } { } { } { } { } { } { } { }
347     {
348         \msg_expandable_error:nnf { cmd } { default-loop }
349         { \exp_args:Nf \tl_trim_spaces:n { \token_to_str:N #4 } }
350         \use_iv:nnnn
351     }
352     \q_stop
353 }
354 \cs_new:Npn \__cmd_end_expandable_defaults:nnnNNn #1#2#3#4#5#6
355 {
356     #6
357     \str_if_eq:nnTF {#1} {#2}
358     { \use_i_delimit_by_q_stop:nw { \group_align_safe_end: #5 #1 } }
359     {
360         \exp_args:No \__cmd_tl_mapthread_function:nnN
361         { #4 #1 } {#3}
362         \__cmd_end_expandable_defaults:nnw
363         \__cmd_end_expandable_defaults:nnnNNn { } {#1} {#3} #4 #5
364     }
365 }
366 \cs_new:Npn \__cmd_end_expandable_defaults:nnw #1#2
367 {
368     \tl_if_novalue:nTF {#2}
369     { \exp_args:No \__cmd_end_expandable_defaults:nw {#1} }

```

```

370      { \__cmd_end_expandable_defaults:nw {#2} }
371    }
372 \cs_new:Npn \__cmd_end_expandable_defaults:nw
373   #1#2 \__cmd_end_expandable_defaults:nnnNNn #3
374   { #2 \__cmd_end_expandable_defaults:nnnNNn { #3 {#1} } }

```

(End definition for `__cmd_end_expandable:NNw` and others.)

1.4 Normalizing the argument specifications

The goal here is to expand aliases and check that the argument specification is valid before the main parsing run. If it is not valid the entire set up is abandoned to avoid any strange internal errors. A function is provided for each argument type that will grab any extra data items and call the loop function after performing the following checks and tasks.

- Check that each argument has the correct number of data items associated with it, and that where a single character is required, one has actually been supplied.
- Check that processors and the markers + and ! are followed by an argument for which they make sense, and are not redundant.
- Check the absence of forbidden types for expandable commands, namely G/v always, and l/u after optional arguments (`xparse` may have inserted braces due to a failed search for an optional argument).
- Check that no optional argument is followed by a mandatory argument with the same delimiter, as otherwise the optional argument could never be omitted.
- Keep track in `\l__cmd_some_long_bool` and `\l__cmd_some_short_bool` of whether the command has some long/short arguments.
- Keep track in `\l__cmd_grab_expandably_bool` of whether all arguments are m/l/u type and short arguments appear before long ones, in which case they can be grabbed expandably just as safely as they could be grabbed nonexpandably. Regardless of that, arguments of expandable commands will be grabbed expandably and arguments of environments will not (because the list of arguments built by non-expandable grabbing is used to pass them to the end-environment code).

Further checks happen at the end of the loop:

- that there are at most 9 arguments;
- that an expandable command does not end with an optional argument (this case is detected by using the fact that `\l__cmd_last_delimiters_tl` is cleared by every mandatory argument and filled by every optional argument).

```

\__cmd_normalize_arg_spec:n Loop through the argument specification, calling an auxiliary specific to each argument
\__cmd_normalize_arg_spec_loop:n type. If any argument is unknown stop the definition.

375 \cs_new_protected:Npn \__cmd_normalize_arg_spec:n #1
376   {
377     \int_zero:N \l__cmd_current_arg_int
378     \tl_clear:N \l__cmd_last_delimiters_tl
379     \tl_clear:N \l__cmd_arg_spec_tl

```

```

380   \bool_set_true:N \l__cmd_grab_expandably_bool
381   \bool_set_false:N \l__cmd_obey_spaces_bool
382   \bool_set_false:N \l__cmd_long_bool
383   \bool_set_false:N \l__cmd_some_obey_spaces_bool
384   \bool_set_false:N \l__cmd_some_long_bool
385   \bool_set_false:N \l__cmd_some_short_bool
386   \__cmd_normalize_arg_spec_loop:n #1
387     \q_recursion_tail \q_recursion_tail \q_recursion_tail \q_recursion_stop
388   \int_compare:nNnT \l__cmd_current_arg_int > 9
389   {
390     \msg_error:nnxx { cmd } { too-many-args }
391     { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
392     \__cmd_bad_def:wn
393   }
394   \bool_if:NT \l__cmd_expandable_bool
395   {
396     \tl_if_empty:NF \l__cmd_last_delimiters_tl
397     {
398       \msg_error:nnxx { cmd } { expandable-ending-optional }
399       { \iow_char:N \\ \l__cmd_function_tl } { \tl_to_str:n {#1} }
400       \__cmd_bad_def:wn
401     }
402   }
403   \bool_if:NT \l__cmd_expandable_bool
404     { \bool_set_true:N \l__cmd_grab_expandably_bool }
405   \bool_if:NT \l__cmd_environment_bool
406     { \bool_set_false:N \l__cmd_grab_expandably_bool }
407   }
408 \cs_new_protected:Npn \__cmd_normalize_arg_spec_loop:n #1
409   {
410     \quark_if_recursion_tail_stop:n {#1}
411     \int_incr:N \l__cmd_current_arg_int
412     \cs_if_exist_use:cF { \__cmd_normalize_type_ \tl_to_str:n {#1} :w }
413     {
414       \bool_lazy_any:nTF
415       {
416         { \str_if_eq_p:nn {#1} { G } }
417         { \str_if_eq_p:nn {#1} { g } }
418         { \str_if_eq_p:nn {#1} { l } }
419         { \str_if_eq_p:nn {#1} { u } }
420       }
421       {
422         \msg_error:nnxx { cmd } { xparse-arg-type }
423         { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
424       }
425       {
426         \msg_error:nnxx { cmd } { unknown-argument-type }
427         { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
428       }
429       \__cmd_bad_def:wn
430     }
431   }

```

(End definition for `__cmd_normalize_arg_spec:n` and `__cmd_normalize_arg_spec_loop:n`.)

`__cmd_normalize_type_d:w` These argument types are aliases of more general ones, for example with the default argument `-NoValue-`. To easily insert that marker expanded in the definitions we call `__cmd_tmp:w` with the argument `-NoValue-`. For argument types that need additional data, check that the data is present (not `\q_recursion_tail`) before proceeding.

```

432 \cs_set_protected:Npn \__cmd_tmp:w #1
433   {
434     \cs_new_protected:Npn \__cmd_normalize_type_d:w ##1##2
435     {
436       \quark_if_recursion_tail_stop_do:nn {##2} { \__cmd_bad_arg_spec:wn }
437       \__cmd_normalize_type_D:w {##1} {##2} {#1}
438     }
439     \cs_new_protected:Npn \__cmd_normalize_type_e:w ##1
440     {
441       \quark_if_recursion_tail_stop_do:nn {##1} { \__cmd_bad_arg_spec:wn }
442       \__cmd_normalize_type_E:w {##1} { }
443     }
444     \cs_new_protected:Npn \__cmd_normalize_type_o:w
445     {
446       \__cmd_normalize_type_D:w [ ] {#1}
447     }
448     \cs_new_protected:Npn \__cmd_normalize_type_0:w
449     {
450       \__cmd_normalize_type_D:w [ ]
451     }
452     \cs_new_protected:Npn \__cmd_normalize_type_r:w ##1##2
453     {
454       \quark_if_recursion_tail_stop_do:nn {##2} { \__cmd_bad_arg_spec:wn }
455       \__cmd_normalize_type_R:w {##1} {##2} {#1}
456     }
457     \cs_new_protected:Npn \__cmd_normalize_type_s:w
458     {
459       \__cmd_normalize_type_t:w *
460     }
461   }
462 \exp_args:No \__cmd_tmp:w { \c_novalue_tl }

```

(End definition for `__cmd_normalize_type_d:w` and others.)

`__cmd_normalize_type_>:w` Check that these prefixes have arguments, namely that the next token is not `\q_recursion_tail`, and remember to leave it after the looping macro. Processors are forbidden in expandable commands. If all is good, store the prefix in the cleaned up `\l__cmd_arg_spec_tl`, and decrement the argument number as prefixes do not correspond to arguments.

```

463 \cs_new_protected:cpx { __cmd_normalize_type_>:w } #1#2
464   {
465     \quark_if_recursion_tail_stop_do:nn {#2} { \__cmd_bad_arg_spec:wn }
466     \bool_if:NT \l__cmd_expandable_bool
467     {
468       \msg_error:nnnx { cmd } { processor-in-expandable }
469       {
470         \iow_char:N \\ \l__cmd_function_tl } { \tl_to_str:n {#1} }
471         \__cmd_bad_def:wn
472       }
473     \tl_put_right:Nx \l__cmd_arg_spec_tl { > { \tl_trim_spaces:n {#1} } }
474     \int_decr:N \l__cmd_current_arg_int
475     \bool_set_false:N \l__cmd_grab_expandably_bool
476     \__cmd_normalize_arg_spec_loop:n {#2}
477   }
478 \cs_new_protected:cpx { __cmd_normalize_type_+:w } #1
479   {

```

```

473 \quark_if_recursion_tail_stop_do:nn {#1} { \__cmd_bad_arg_spec:wn }
474 \bool_if:NT \l__cmd_long_bool
475 {
476     \msg_error:nnnx { cmd } { two-markers }
477     { \__cmd_environment_or_command: } { + }
478     \__cmd_bad_def:wn
479 }
480 \bool_set_true:N \l__cmd_long_bool
481 \int_decr:N \l__cmd_current_arg_int
482 \__cmd_normalize_arg_spec_loop:n {#1}
483 }
484 \cs_new_protected:cpn { __cmd_normalize_type_!:@w } #1
485 {
486     \quark_if_recursion_tail_stop_do:nn {#1} { \__cmd_bad_arg_spec:wn }
487     \bool_if:NT \l__cmd_obey_spaces_bool
488     {
489         \msg_error:nnnx { cmd } { two-markers }
490         { \__cmd_environment_or_command: } { ! }
491         \__cmd_bad_def:wn
492     }
493     \bool_set_true:N \l__cmd_obey_spaces_bool
494     \bool_set_true:N \l__cmd_some_obey_spaces_bool
495     \int_decr:N \l__cmd_current_arg_int
496     \__cmd_normalize_arg_spec_loop:n {#1}
497 }

```

(End definition for `__cmd_normalize_type_>:w`, `__cmd_normalize_type_+:w`, and `__cmd_normalize_type_!:@w`.)

`__cmd_normalize_type_D:w` Optional argument types. Check that all required data is present (and consists of single characters if applicable) and check for forbidden types for expandable commands. For E-type require that there is at least one embellishment, that each one is a single character, and that there aren't more optional arguments than embellishments; also remember that each embellishment counts as one argument for `\l__cmd_current_arg_int`. Then in each case store the data in `\l__cmd_arg_spec_tl`, and for later checks store in `\l__cmd_last_delimiters_tl` the tokens whose presence determines whether there is an optional argument (for braces store {}, seen later as an empty delimiter).

```

498 \cs_new_protected:Npn \__cmd_normalize_type_D:w #1#2#3
499 {
500     \quark_if_recursion_tail_stop_do:nn {#3} { \__cmd_bad_arg_spec:wn }
501     \__cmd_single_token_check:n {#1} \__cmd_allowed_token_check:N #1
502     \__cmd_single_token_check:n {#2}
503     \__cmd_add_arg_spec:n { D #1 #2 {#3} }
504     \tl_put_right:Nn \l__cmd_last_delimiters_tl {#1}
505     \bool_set_false:N \l__cmd_grab_expandably_bool
506     \__cmd_normalize_arg_spec_loop:n
507 }
508 \cs_new_protected:Npn \__cmd_normalize_type_E:w #1#2
509 {
510     \quark_if_recursion_tail_stop_do:nn {#2} { \__cmd_bad_arg_spec:wn }
511     \tl_if_blank:nT {#1} { \__cmd_bad_arg_spec:wn }
512     \tl_map_function:nN {#1} \__cmd_single_token_check:n
513     \tl_map_function:nN {#1} \__cmd_allowed_token_check:N
514     \__cmd_normalize_E_unique_check:w #1 \q_nil \q_stop

```

```

515     \int_compare:nNnT { \tl_count:n {#2} } > { \tl_count:n {#1} }
516     { \__cmd_bad_arg_spec:wn }
517     \__cmd_add_arg_spec:n { E {#1} {#2} }
518     \tl_put_right:Nn \l__cmd_last_delimiters_tl {#1}
519     \bool_set_false:N \l__cmd_grab_expandably_bool
520     \int_add:Nn \l__cmd_current_arg_int { \tl_count:n {#1} - 1 }
521     \__cmd_normalize_arg_spec_loop:n
522   }
523 \cs_new_protected:Npn \__cmd_normalize_E_unique_check:w #1#2 \q_stop
524   {
525     \quark_if_nil:NF #1
526     {
527       \tl_if_in:nnT {#2} {#1} { \__cmd_bad_arg_spec:wn }
528       \__cmd_normalize_E_unique_check:w #2 \q_stop
529     }
530   }
531 \cs_new_protected:Npn \__cmd_normalize_type_t:w #1
532   {
533     \quark_if_recursion_tail_stop_do:Nn #1 { \__cmd_bad_arg_spec:wn }
534     \__cmd_single_token_check:n {#1} \__cmd_allowed_token_check:N #1
535     \tl_put_right:Nx \l__cmd_arg_spec_tl
536     {
537       \bool_if:NT \l__cmd_obey_spaces_bool { ! }
538       t \exp_not:n {#1}
539     }
540     \tl_put_right:Nn \l__cmd_last_delimiters_tl {#1}
541     \bool_set_false:N \l__cmd_grab_expandably_bool
542     \bool_set_false:N \l__cmd_obey_spaces_bool
543     \bool_set_false:N \l__cmd_long_bool
544     \__cmd_normalize_arg_spec_loop:n
545   }

```

(End definition for `__cmd_normalize_type_D:w` and others.)

`__cmd_normalize_type_m:w` Mandatory arguments. First check the required data is present, consists of single characters where applicable, and that the argument type is allowed for expandable commands if applicable. For the `m` and `R` argument types check that they do not follow some optional argument with that delimiter as otherwise the optional argument could not be omitted. Then save data in `\l__cmd_arg_spec_tl`, count the mandatory argument, and empty the list of last delimiters.

```

546 \cs_new_protected:Npn \__cmd_normalize_type_m:w
547   {
548     \__cmd_delimiter_check:nnn { } { m } { \iow_char:N \{ }
549     \__cmd_add_arg_spec_mandatory:n { m }
550     \__cmd_normalize_arg_spec_loop:n
551   }
552 \cs_new_protected:Npn \__cmd_normalize_type_R:w #1#2#3
553   {
554     \quark_if_recursion_tail_stop_do:nn {#3} { \__cmd_bad_arg_spec:wn }
555     \__cmd_single_token_check:n {#1} \__cmd_allowed_token_check:N #1
556     \__cmd_single_token_check:n {#2}
557     \__cmd_delimiter_check:nnn {#1} { R/r } { \tl_to_str:n {#1} }
558     \bool_set_false:N \l__cmd_grab_expandably_bool
559     \__cmd_add_arg_spec_mandatory:n { R #1 #2 {#3} }

```

```

560      \_\_cmd\_normalize\_arg\_spec\_loop:n
561  }
562 \cs_new_protected:Npn \_\_cmd\_normalize\_type\_v:w
563 {
564     \_\_cmd\_normalize\_check\_gv:N v
565     \_\_cmd\_add\_arg\_spec\_mandatory:n { v }
566     \_\_cmd\_normalize\_arg\_spec\_loop:n
567 }

```

(End definition for __cmd_normalize_type_m:w, __cmd_normalize_type_R:w, and __cmd_normalize_type_v:w.)

__cmd_normalize_type_b:w This argument type is not allowed for commands. This is only allowed at the end of the argument specification, hence we check that #1 is the end.

```

568 \cs_new_protected:Npn \_\_cmd\_normalize\_type\_b:w #1
569 {
570     \bool_if:NF \l__cmd_environment_bool
571     {
572         \msg_error:nnxx { cmd } { invalid-command-arg }
573         { \_\_cmd_environment_or_command: } { b }
574         \_\_cmd_bad_def:wn
575     }
576     \tl_clear:N \l__cmd_last_delimiters_tl
577     \_\_cmd_add_arg_spec:n { b }
578     \quark_if_recursion_tail_stop:n {#1}
579     \msg_error:nnxx { cmd } { arg-after-body }
580     { \_\_cmd_environment_or_command: }
581     { \tl_to_str:n {#1} }
582     \_\_cmd_bad_def:wn
583 }

```

(End definition for __cmd_normalize_type_b:w.)

__cmd_single_token_check:n Checks that the argument is a single (non-space) token (possibly surrounded by spaces), and aborts the definition otherwise.

```

584 \cs_new_protected:Npn \_\_cmd_single_token_check:n #1
585 {
586     \tl_trim_spaces_apply:nN {#1} \tl_if_single_token:nF
587     {
588         \msg_error:nnxx { cmd } { not-single-token }
589         { \_\_cmd_environment_or_command: } { \tl_to_str:n {#1} }
590         \_\_cmd_bad_def:wn
591     }
592 }

```

(End definition for __cmd_single_token_check:n.)

__cmd_allowed_token_check:N Some tokens are now allowed as delimiters for some argument types, notably implicit begin/end-group tokens (\bgroup/\egroup). The major problem with these tokens is that for \peek_... functions, a literal {₁. is virtually indistinguishable from a \bgroup or other token which was \let to a {₁, and the same goes for }₂. All other tokens can be easily distinguished from their implicit counterparts by grabbing them and looking at the string length (see __cmd_token_if_cs:NTF), but for begin/end group tokens

that is not possible without the risk of mistakenly grabbing the entire brace group (potentially leading to a ! Runaway argument error) or trying to grab a $\}$, leading to an ! Argument of \dots has an extra } error.

```

593 \cs_new_protected:Npn \__cmd_allowed_token_check:N #1
594   {
595     \token_if_eq_meaning:NNTF #1 \c_group_begin_token
596       { \use:n }
597       {
598         \token_if_eq_meaning:NNTF #1 \c_group_end_token
599           { \use:n }
600           { \use_none:n }
601       }
602   {
603     \msg_error:nnxx { cmd } { forbidden-group-token }
604     { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
605     {
606       \token_if_eq_meaning:NNTF #1 \c_group_begin_token
607         { begin } { end }
608       }
609     \__cmd_bad_def:wn
610   }
611 }
```

(End definition for __cmd_allowed_token_check:N.)

__cmd_normalize_check_gv:N
__cmd_normalize_check_lu:N Called for arguments that are always forbidden, or forbidden after an optional argument, for expandable commands.

```

612 \cs_new_protected:Npn \__cmd_normalize_check_gv:N #1
613   {
614     \bool_if:NT \l__cmd_expandable_bool
615     {
616       \msg_error:nnxx { cmd } { invalid-expandable-arg }
617       { \iow_char:N \\ \l__cmd_function_tl } { \tl_to_str:n {#1} }
618       \__cmd_bad_def:wn
619     }
620     \bool_set_false:N \l__cmd_grab_expandably_bool
621   }
622 \cs_new_protected:Npn \__cmd_normalize_check_lu:N #1
623   {
624     \bool_if:NT \l__cmd_expandable_bool
625     {
626       \tl_if_empty:NF \l__cmd_last_delimiters_tl
627       {
628         \msg_error:nnxx { cmd } { invalid-after-optional-expandably }
629         { \iow_char:N \\ \l__cmd_function_tl } { \tl_to_str:n {#1} }
630         \__cmd_bad_def:wn
631       }
632     }
633 }
```

(End definition for __cmd_normalize_check_gv:N and __cmd_normalize_check_lu:N.)

__cmd_delimiter_check:nnn Called for m and R arguments. Checks that the leading token does not coincide with the token denoting the presence of a previous optional argument. Instead of dealing with braces for the m-type we use an empty delimiter to denote that case.

```

634 \cs_new_protected:Npn \__cmd_delimiter_check:n {#1} {#3}
635   {
636     \tl_map_inline:Nn \l__cmd_last_delimiters_tl
637     {
638       \tl_if_eq:nnT {##1} {#1}
639       {
640         \msg_warning:nnxx { cmd } { optional-mandatory }
641         {#2} {#3}
642       }
643     }
644   }

```

(End definition for `__cmd_delimiter_check:n`.)

`__cmd_bad_arg_spec:wn`
`__cmd_bad_def:wn`

If the argument specification is wrong, this provides an escape from the entire definition process.

```

645 \cs_new_protected:Npn \__cmd_bad_arg_spec:wn {#1} \__cmd_break_point:n {#2}
646   {
647     \msg_error:nnxx { cmd } { bad-arg-spec }
648     { \__cmd_environment_or_command: } { \tl_to_str:n {#2} }
649   }
650 \cs_new_protected:Npn \__cmd_bad_def:wn {#1} \__cmd_break_point:n {#2} { }

```

(End definition for `__cmd_bad_arg_spec:wn` and `__cmd_bad_def:wn`.)

`__cmd_add_arg_spec:n`
`__cmd_add_arg_spec_mandatory:n`

When adding an argument to the argument specification, set the `some_long` or `some_short` booleans as appropriate and clear the booleans keeping track of + and ! markers. Before that, test for a short argument following some long arguments: this is forbidden for expandable commands and prevents grabbing arguments expandably.

For mandatory arguments do some more work, in particular complain if they were preceded by !.

```

651 \cs_new_protected:Npn \__cmd_add_arg_spec:n {#1}
652   {
653     \bool_lazy_and:nnT
654     { ! \l__cmd_long_bool }
655     { \l__cmd_some_long_bool }
656   {
657     \bool_if:NT \l__cmd_expandable_bool
658     {
659       \msg_error:nnx { cmd } { long-short-mix }
660       { \iow_char:N \\ \l__cmd_function_tl }
661       \__cmd_bad_def:wn
662     }
663     \bool_set_false:N \l__cmd_grab_expandably_bool
664   }
665 \bool_if:NTF \l__cmd_long_bool
666   { \bool_set_true:N \l__cmd_some_long_bool }
667   { \bool_set_true:N \l__cmd_some_short_bool }
668 \tl_put_right:Nx \l__cmd_arg_spec_tl
669   {
670     \bool_if:NT \l__cmd_long_bool { + }
671     \bool_if:NT \l__cmd_obey_spaces_bool { ! }
672     \exp_not:n {#1}
673   }

```

```

674      \bool_set_false:N \l__cmd_long_bool
675      \bool_set_false:N \l__cmd_obey_spaces_bool
676    }
677  \cs_new_protected:Npn \__cmd_add_arg_spec_mandatory:n #1
678  {
679    \bool_if:NT \l__cmd_some_obey_spaces_bool
680    {
681      \msg_error:nnnx { cmd } { invalid-bang }
682      { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
683      \__cmd_bad_def:wn
684    }
685    \tl_clear:N \l__cmd_last_delimiters_tl
686    \__cmd_add_arg_spec:n {#1}
687  }

```

(End definition for `__cmd_add_arg_spec:n` and `__cmd_add_arg_spec_mandatory:n`.)

1.5 Preparing the signature: general mechanism

`__cmd_prepare_signature:n`
`__cmd_prepare_signature:N`
`__cmd_prepare_signature_bypass:N`

Actually creating the signature uses the same loop approach as normalizing the signature. There are first a number of variables which need to be set to track what is going on. Many of these variables are unused when defining expandable commands.

```

688 \cs_new_protected:Npn \__cmd_prepare_signature:n #1
689  {
690    \int_zero:N \l__cmd_current_arg_int
691    \bool_set_false:N \l__cmd_long_bool
692    \bool_set_false:N \l__cmd_obey_spaces_bool
693    \int_zero:N \l__cmd_m_args_int
694    \bool_set_false:N \l__cmd_defaults_bool
695    \tl_clear:N \l__cmd_defaults_tl
696    \tl_clear:N \l__cmd_process_all_tl
697    \tl_clear:N \l__cmd_process_one_tl
698    \bool_set_false:N \l__cmd_process_some_bool
699    \tl_clear:N \l__cmd_signature_tl
700    \__cmd_prepare_signature:N #1 \q_recursion_tail \q_recursion_stop
701    \bool_if:NF \l__cmd_expandable_bool { \__cmd_flush_m_args: }
702  }

```

The main looping function does not take an argument, but carries out the reset on the processor boolean. This is split off from the rest of the process so that when actually setting up processors the flag-reset can be bypassed.

For each known argument type there is an appropriate function to actually do the addition to the signature. These are separate for expandable and standard functions, as the approaches are different.

```

703 \cs_new_protected:Npn \__cmd_prepare_signature:N
704  {
705    \bool_set_false:N \l__cmd_prefixed_bool
706    \__cmd_prepare_signature_bypass:N
707  }
708 \cs_new_protected:Npn \__cmd_prepare_signature_bypass:N #1
709  {
710    \quark_if_recursion_tail_stop:N #1
711    \use:c
712    {

```

```

713      __cmd_add
714      \bool_if:NT \l__cmd_grab_expandably_bool { _expandable }
715      _type_ \token_to_str:N #1 :w
716    }
717  }

(End definition for \__cmd_prepare_signature:n, \__cmd_prepare_signature:N, and
\__cmd_prepare_signature_bypass:N.)

```

1.6 Setting up a standard signature

Each argument-adding function appends to the signature a grabber (and for some types, the delimiters or default value), except the one for `m` arguments. These are collected and added to the signature all at once by `__cmd_flush_m_args:`, called for every other argument type. All of the functions then call the loop function `__cmd_prepare_signature:N`. Default values of arguments are collected by `__cmd_add_default:n` rather than being stored with the argument; this function and `__cmd_add_default:` are also responsible for keeping track of `\l__cmd_current_arg_int`.

`__cmd_add_type_+::w` Making the next argument long means setting the flag. The `m` arguments are recorded here as this has to be done for every case where there is then a long argument.

```

718 \cs_new_protected:cpn { __cmd_add_type_+::w }
719 {
720   __cmd_flush_m_args:
721   \bool_set_true:N \l__cmd_long_bool
722   \bool_set_true:N \l__cmd_prefixed_bool
723   \__cmd_prepare_signature_bypass:N
724 }

```

(End definition for __cmd_add_type_+::w.)

`__cmd_add_type_!::w` Much the same for controlling trailing optional arguments.

```

725 \cs_new_protected:cpn { __cmd_add_type_!::w }
726 {
727   __cmd_flush_m_args:
728   \bool_set_true:N \l__cmd_obey_spaces_bool
729   \bool_set_true:N \l__cmd_prefixed_bool
730   \__cmd_prepare_signature_bypass:N
731 }

```

(End definition for __cmd_add_type_!::w.)

`__cmd_add_type_>::w` When a processor is found, the processor code is stored. It will be used by `__cmd_args_process:` once arguments are all found. Here too the loop calls `__cmd_prepare_signature_bypass:N` rather than `__cmd_prepare_signature:N` so that the flag is not reset.

```

732 \cs_new_protected:cpn { __cmd_add_type_>::w } #1
733 {
734   __cmd_flush_m_args:
735   \bool_set_true:N \l__cmd_prefixed_bool
736   \bool_set_true:N \l__cmd_process_some_bool
737   \tl_put_left:Nn \l__cmd_process_one_tl { {#1} }
738   \__cmd_prepare_signature_bypass:N
739 }

```

(End definition for `__cmd_add_type_>:w.`)

```
\_\_cmd\_add\_type\_b:w
740 \cs_new_protected:Npn \_\_cmd\_add\_type\_b:w
741 {
742     \_\_cmd\_flush_m\_args:
743     \_\_cmd\_add\_default:
744     \_\_cmd\_add\_grabber:N b
745     \_\_cmd\_prepare\_signature:N
746 }
```

(End definition for `__cmd_add_type_b:w.`)

```
\_\_cmd\_add\_type\_D:w
747 \cs_new_protected:Npn \_\_cmd\_add\_type\_D:w #1#2#3
748 {
749     \_\_cmd\_flush_m\_args:
750     \_\_cmd\_add\_default:n {#3}
751     \_\_cmd\_add\_grabber:N D
752     \tl_put_right:Nn \l_\_cmd\_signature_tl { #1 #2 }
753     \_\_cmd\_prepare\_signature:N
754 }
```

(End definition for `__cmd_add_type_D:w.`)

`__cmd_add_type_E:w` The E-type argument needs a special handling of default values. Since each embellishment is a separate argument, it also needs to replicate the argument processors for each embellishment argument so that the numbers of arguments and processors remain in sync.

```
755 \cs_new_protected:Npn \_\_cmd\_add\_type\_E:w #1#2
756 {
757     \_\_cmd\_flush_m\_args:
758     \_\_cmd\_add\_default_E:nn {#1} {#2}
759     \use:x
760     {
761         \_\_cmd\_replicate_processor:nn { \tl_count:n {#1} }
762         { \exp_not:o \l_\_cmd\_process\_one_tl }
763     }
764     \_\_cmd\_add\_grabber:N E
765     \tl_put_right:Nn \l_\_cmd\_signature_tl { {#1} }
766     \_\_cmd\_prepare\_signature:N
767 }
```

(End definition for `__cmd_add_type_E:w.`)

`__cmd_replicate_processor:nn` In the command's argument processor signature (the final argument of `__cmd_start:nNNnnn`) there is one braced item for each formal argument (up to nine), and in each of these items there is one braced item for each processor (as many as there were processors declared for a given argument). Something like this:

```
{ % argument processors
{ % argument 1
{ processor 1 } { processor 2 } ... { processor n }
```

```

} % end argument 1
{ ... } % argument 2
:
{ ... } % argument n
} % end argument processors

```

The function `__cmd_add_grabber:N` adds one single grabber for an argument, and adds the braced item for that one argument. However, in an E-type argument each embellishment requires its own formal argument, so we need to break out of one layer of braces in `\l__cmd_process_one_tl`, add copies of the processor as necessary, and then return the removed brace. The function below does just that: it defines `\l__cmd_process_one_tl` starting with a `}_2` and ending with a `{_1`, so that it adds as many processors as needed when x-expanded.

```

768 \cs_new_protected:Npn \__cmd_replicate_processor:nn #1 #2
769  {
770    \int_compare:nNnF {#1} > { 1 } { \use_none:nnn }
771    \tl_set:Nx \l__cmd_process_one_tl
772    {
773      \exp_not:n { \exp_not:n {#2} \if_false: { \fi: } }
774      \prg_replicate:nn { #1 - 2 }
775      { \exp_not:n { \exp_not:n { {#2} } } }
776      \exp_not:n { { \if_false: } \fi: \exp_not:n {#2} }
777    }
778  }

```

(End definition for `__cmd_replicate_processor:nn`.)

`__cmd_add_type_m:w` The `m` type is special as short arguments which are not post-processed are simply counted at this stage. Thus there is a check to see if either of these cases apply. If so, a one-argument grabber is added to the signature. On the other hand, if a standard short argument is required it is simply counted at this stage, to be added later using `__cmd_flush_m_args:`.

```

779 \cs_new_protected:Npn \__cmd_add_type_m:w
780  {
781    \__cmd_add_default:
782    \bool_if:NTF \l__cmd_prefixed_bool
783      { \__cmd_add_grabber:N m }
784      { \int_incr:N \l__cmd_m_args_int }
785    \__cmd_prepare_signature:N
786  }

```

(End definition for `__cmd_add_type_m:w`.)

`__cmd_add_type_R:w` The R-type argument is very similar to the D-type.

```

787 \cs_new_protected:Npn \__cmd_add_type_R:w #1#2#3
788  {
789    \__cmd_flush_m_args:
790    \__cmd_add_default:n {#3}
791    \__cmd_add_grabber:N R
792    \tl_put_right:Nn \l__cmd_signature_tl { #1 #2 }
793    \__cmd_prepare_signature:N
794  }

```

(End definition for `__cmd_add_type_R:w.`)

`__cmd_add_type_t:w` Setting up a `t` argument means collecting one token for the test, and adding it along with the grabber to the signature.

```
795 \cs_new_protected:Npn \_\_cmd\_add\_type\_t:w #1
796 {
797     \_\_cmd\_flush_m\_args:
798     \_\_cmd\_add\_default:
799     \_\_cmd\_add\_grabber:N t
800     \tl_put_right:Nn \l_\_\_cmd\_signature_tl {\#1}
801     \_\_cmd\_prepare\_signature:N
802 }
```

(End definition for `__cmd_add_type_t:w.`)

`__cmd_add_type_v:w` At this stage, the `v` argument is identical to `l` except that since the grabber may fail to read a verbatim argument we need a default value.

```
803 \cs_new_protected:Npn \_\_cmd\_add\_type\_v:w
804 {
805     \_\_cmd\_flush_m\_args:
806     \exp_args:No \_\_cmd\_add\_default:n \c_novalue_tl
807     \_\_cmd\_add\_grabber:N v
808     \_\_cmd\_prepare\_signature:N
809 }
```

(End definition for `__cmd_add_type_v:w.`)

`__cmd_flush_m_args:` As `m` arguments are simply counted, there is a need to add them to the token register in a block. As this function can only be called if something other than `m` turns up, the flag can be switched here.

```
810 \cs_new_protected:Npn \_\_cmd\_flush_m\_args:
811 {
812     \int_compare:nNnT \l_\_\_cmd_m\_args_int > 0
813     {
814         \tl_put_right:Nx \l_\_\_cmd\_signature_tl
815         { \exp_not:c { \_\_cmd\_grab_m\_ \int_use:N \l_\_\_cmd_m\_args_int :w } }
816         \tl_put_right:Nx \l_\_\_cmd\_process\_all_tl
817         { \prg_replicate:nn { \l_\_\_cmd_m\_args_int } { { } } }
818     }
819     \int_zero:N \l_\_\_cmd_m\_args_int
820 }
```

(End definition for `__cmd_flush_m_args:.`)

`__cmd_add_grabber:N` To keep the various checks needed in one place, adding the grabber to the signature is done here. The only questions are whether the grabber should be long or not, and whether to obey spaces. The `\l___cmd_obey_spaces_bool` boolean can only be `true` for trailing optional arguments. In that case spaces will not be ignored when looking for that optional argument.

```
821 \cs_new_protected:Npn \_\_cmd\_add\_grabber:N #1
822 {
823     \tl_put_right:Nx \l_\_\_cmd\_signature_tl
824     {
825         \exp_not:c
```

```

826         {
827             __cmd_grab_ #1
828             \bool_if:NT \l__cmd_long_bool { _long }
829             \bool_if:NT \l__cmd_obey_spaces_bool { _obey_spaces }
830             :w
831         }
832     }
833     \bool_set_false:N \l__cmd_long_bool
834     \bool_set_false:N \l__cmd_obey_spaces_bool
835     \tl_put_right:Nx \l__cmd_process_all_tl
836     {
837         {
838             \if_charcode:w E #1 \use_i:nn \fi:
839             \exp_not:o \l__cmd_process_one_tl
840         }
841     }
842     \tl_clear:N \l__cmd_process_one_tl
843 }

```

(End definition for `__cmd_add_grabber:N`.)

`__cmd_add_default:n`
`__cmd_add_default:`
`__cmd_add_default_E:nn` Store the default value of an argument, or rather code that gives that default value (it may involve other arguments). This is `\c_novalue_tl` for arguments with no actual default or with default `-NoValue-`; and (in a brace group) `\prg_do_nothing:` followed by a default value for others. For E-type arguments, pad the defaults `#2` with some `\c_novalue_tl` until there are as many as embellishments `#1`. These functions are also used when defining expandable commands.

```

844 \cs_new_protected:Npn \__cmd_add_default:n #1
845   {
846     \tl_if_novalue:nTF {#1}
847       { \__cmd_add_default: }
848       {
849           \int_incr:N \l__cmd_current_arg_int
850           \bool_set_true:N \l__cmd_defaults_bool
851           \tl_put_right:Nn \l__cmd_defaults_tl { { \prg_do_nothing: #1 } }
852       }
853   }
854 \cs_new_protected:Npn \__cmd_add_default:
855   {
856     \int_incr:N \l__cmd_current_arg_int
857     \tl_put_right:Nn \l__cmd_defaults_tl { \c_novalue_tl }
858   }
859 \cs_new_protected:Npn \__cmd_add_default_E:nn #1#2
860   {
861     \tl_map_function:nN {#2} \__cmd_add_default:n
862     \prg_replicate:nn
863       { \tl_count:n {#1} - \tl_count:n {#2} }
864       { \__cmd_add_default: }
865   }

```

(End definition for `__cmd_add_default:n`, `__cmd_add_default:`, and `__cmd_add_default_E:nn`.)

1.7 Setting up expandable types

The approach here is not dissimilar to that for standard types, but fewer types are supported. There is also a need to define the per-function auxiliaries: this is done here, while the general grabbers are dealt with later.

```
\_\_cmd\_add\_expandable\_type\_+:\w
We have already checked that short arguments are before long arguments, so \l\_\_cmd\_long\_bool only changes from false to true once (and there is no need to reset it after each argument). Continue the loop.
```

```
866 \cs_new_protected:cpn { __cmd_add_expandable_type_+:\w }
867   {
868     \bool_set_true:N \l\_\_cmd_long_bool
869     \_\_cmd_prepare_signature:N
870   }
```

(End definition for __cmd_add_expandable_type_+:\w.)

The set up for D-type arguments involves constructing a rather complex auxiliary which is used repeatedly when grabbing. There is an auxiliary here so that the R-type can share code readily: #1 is D or R. The `_aux:NN` auxiliary is needed if the two delimiting tokens are identical: in contrast to the non-expandable route, the grabber here has to act differently for this case.

```
871 \cs_new_protected:Npn \_\_cmd\_add\_expandable\_type\_D:\w
872   { __cmd_add_expandable_type_D_aux:NNNn D }
873 \cs_new_protected:Npn \_\_cmd\_add\_expandable\_type\_D\_aux:NNNn #1#2#3#4
874   {
875     \_\_cmd\_add\_default:n {#4}
876     \tl_if_eq:nnTF {#2} {#3}
877       { __cmd_add_expandable_type_D_aux:NN #1 #2 }
878       { __cmd_add_expandable_type_D_aux:NNN #1 #2 #3 }
879     \_\_cmd_prepare_signature:N
880   }
881 \cs_new_protected:Npn \_\_cmd\_add\_expandable\_type\_D\_aux:NNN #1#2#3
882   {
883     \bool_if:NTF \l\_\_cmd_long_bool
884       { \cs_set:cpx }
885       { \cs_set_nopar:cpx }
886       { \l\_\_cmd_expandable_aux_name_t1 } ##1 ##2 #2 ##3 \q\_\_cmd ##4 #3
887       { ##1 {##2} {##3} {##4} }
888     \_\_cmd\_add\_expandable\_grabber:nn {#1}
889     {
890       \exp_not:c { \l\_\_cmd_expandable_aux_name_t1 }
891       \exp_not:n { #2 #3 }
892     }
893   }
894 \cs_new_protected:Npn \_\_cmd\_add\_expandable\_type\_D\_aux:NN #1#2
895   {
896     \bool_if:NTF \l\_\_cmd_long_bool
897       { \cs_set:cpx }
898       { \cs_set_nopar:cpx }
899       { \l\_\_cmd_expandable_aux_name_t1 } ##1 #2 ##2 #2
900       { ##1 {##2} }
901     \_\_cmd\_add\_expandable\_grabber:nn { #1_alt }
902     {
```

```

903         \exp_not:c { \l__cmd_expandable_aux_name_tl }
904         \exp_not:n {#2}
905     }
906 }

```

(End definition for `__cmd_add_expandable_type_D:w` and others.)

```
\__cmd_add_expandable_type_E:w
\__cmd_add_expandable_type_E_aux:n
```

For each embellishment, use `__cmd_get_grabber>NN` to obtain an auxiliary delimited by that token and store a pair constituted of the auxiliary and the token in `\l__cmd_tmpb_t1`, before appending the whole set of these pairs to the signature, and an equal number of `-NoValue-` markers (regardless of the default values of arguments). Set the current argument appropriately.

```

907 \cs_new_protected:Npn \__cmd_add_expandable_type_E:w #1#2
908 {
909     \__cmd_add_default_E:nn {#1} {#2}
910     \tl_clear:N \l__cmd_tmpb_t1
911     \tl_map_function:nN {#1} \__cmd_add_expandable_type_E_aux:n
912     \__cmd_add_expandable_grabber:nn
913     { E \bool_if:NT \l__cmd_long_bool { _long } }
914     {
915         { \exp_not:o \l__cmd_tmpb_t1 }
916         {
917             \prg_replicate:nn { \tl_count:n {#1} }
918             { { \c_novalue_t1 } }
919         }
920     }
921     \__cmd_prepare_signature:N
922 }
923 \cs_new_protected:Npn \__cmd_add_expandable_type_E_aux:n #1
924 {
925     \__cmd_get_grabber:NN #1 \l__cmd_tmpa_t1
926     \tl_put_right:Nx \l__cmd_tmpb_t1
927     { \exp_not:o \l__cmd_tmpa_t1 \exp_not:N #1 }
928 }
```

(End definition for `__cmd_add_expandable_type_E:w` and `__cmd_add_expandable_type_E_aux:n`.)

```
\__cmd_add_expandable_type_m:w
```

Unlike the standard case, when working expandably each argument is always grabbed separately.

```

929 \cs_new_protected:Npn \__cmd_add_expandable_type_m:w
930 {
931     \__cmd_add_default:
932     \__cmd_add_expandable_grabber:nn
933     { m \bool_if:NT \l__cmd_long_bool { _long } } { }
934     \__cmd_prepare_signature:N
935 }
```

(End definition for `__cmd_add_expandable_type_m:w`.)

```
\__cmd_add_expandable_type_R:w
```

The R-type is very similar to the D-type argument, and so the same internals are used.

```

936 \cs_new_protected:Npn \__cmd_add_expandable_type_R:w
937 { \__cmd_add_expandable_type_D_aux:NNNn R }
```

(End definition for `__cmd_add_expandable_type_R:w`.)

__cmd_add_expandable_type_t:w An auxiliary delimited by #1 is built now. It will be used to test for the presence of that token.

```
938 \cs_new_protected:Npn \_\_cmd\_add\_expandable\_type\_t:w #1
939 {
940     \_\_cmd\_add\_default:
941     \_\_cmd\_get\_grabber:NN #1 \l_\_cmd\_tmpa_tl
942     \_\_cmd\_add\_expandable\_grabber:nn { t }
943     {
944         \exp_not:o \l_\_cmd\_tmpa_tl
945         \exp_not:N #1
946     }
947     \_\_cmd\_prepare\_signature:N
948 }
```

(End definition for __cmd_add_expandable_type_t:w.)

__cmd_add_expandable_grabber:nn This is called for all arguments to place the right grabber in the signature.

```
949 \cs_new_protected:Npn \_\_cmd\_add\_expandable\_grabber:nn #1#2
950 {
951     \tl_put_right:Nx \l_\_cmd\_signature_tl
952     { \exp_not:c { \_\_cmd\_expandable\_grab\_#1 :w } #2 }
953 }
```

(End definition for __cmd_add_expandable_grabber:nn.)

__cmd_get_grabber>NN Given a token #1, defines an expandable function delimited by that token and stores it in the token list #2. The function is named after the token, unless that function name is already taken by some other grabber (this can happen in the rare case where delimiters with different category codes are used in the same document): in that case use a global counter to get a unique name. Since the grabbers are not named after `xparse` commands they should not be used to get material from the input stream.

```
954 \cs_new_protected:Npn \_\_cmd\_get\_grabber:NN #1#2
955 {
956     \cs_set:Npn \_\_cmd\_tmp:w ##1 #1 {##1}
957     \exp_args:Nc \_\_cmd\_get\_grabber\_auxi:NN
958     { \_\_cmd\_grabber\_ \token_to_str:N #1 :w } #2
959 }
960 \cs_new_protected:Npn \_\_cmd\_get\_grabber\_auxi:NN #1#2
961 {
962     \cs_if_eq:NNTF \_\_cmd\_tmp:w #1
963     { \tl_set:Nn #2 {#1} }
964     {
965         \cs_if_exist:NTF #1
966         {
967             \int_gincr:N \g_\_cmd\_grabber_int
968             \exp_args:Nc \_\_cmd\_get\_grabber\_auxi:NN
969             {
970                 \_\_cmd\_grabber_
971                 - \int_use:N \g_\_cmd\_grabber_int :w
972             }
973             #2
974         }
975         { \_\_cmd\_get\_grabber\_auxii:NN #1 #2 }
976     }
```

```

977     }
978 \cs_new_protected:Npn \__cmd_get_grabber_auxii:NN #1#2
979   {
980     \cs_set_eq:NN #1 \__cmd_tmp:w
981     \tl_set:Nn #2 {#1}
982   }
983 (End definition for \__cmd_get_grabber:NN, \__cmd_get_grabber_auxi:NN, and
984   \__cmd_get_grabber_auxii:NN.)
```

1.7.1 Copying a command and its internal structure

```

983 \IfFileExists{2021/11/15}{\__cmd_copy:NN}%
984 \IfFileExists{}{\SupportNewCommandCopy-in-ltcmd}
```

Since the 2020-10-01 L^AT_EX 2_E release, support for copying, and showing the definition of, robust commands has been available, but the specifics of each command are implemented separately. Here we'll add support for copying and showing `ltcmd` definitions.

To fully support copying, we need two commands: a conditional to test if a command is in fact a `ltcmd` command, and another command to actually copy the command. The conditional is defined later as `__kernel_cmd_if_xparse:NTF`, so now to the copying: This macro just branches to the proper copying command by using `__cmd_cmd_type_cases:NnnnnF`. The copying command takes the names of the commands to be copied to and from, and the actual commands as its four arguments.

```

985 \cs_new_protected:Npn \__cmd_copy:NN #1 #2
986   {
987     \use:x
988     {
989       \int_set:Nn \tex_escapechar:D { 92 }
990       \exp_not:N \__cmd_cmd_type_cases:NnnnnF \exp_not:N #2
991       { \__cmd_copy_command:nN } { \__cmd_copy_expandable:nN } { \__cmd_copy_environment:nN } { \__cmd_copy_environment_end:nN } { \__cmd_cant_copy:nw { non-ltcmd } } { \cs_to_str:N #1 } { \cs_to_str:N #2 }
992       \exp_not:N #1 \exp_not:N #2
993       \exp_not:N \__cmd_break_point:n { \cs_to_str:N #2 }
994       \int_set:Nn \tex_escapechar:D { \int_use:N \tex_escapechar:D }
995     }
996   }
997 \cs_new_protected:Npn \__cmd_set_eq_if_exist:NN #1 #2
998   { \cs_if_exist:NTF #2 { \cs_set_eq:NN } { \use_none:n } #1 #2 }
999 \cs_generate_variant:Nn \__cmd_set_eq_if_exist:NN { cc }
```

An utility macro similar to `__cmd_bad_def:wn` to abort a command copy. Contrary to `__cmd_bad_def:wn` though, when this happens the issue is most likely internal, because the command was already (supposedly) correctly defined so it should be copyable. Hopefully this macro will never be used ever, but if it does, apologise and give the reason for the failure so the user can report.

```

1005 \cs_new_protected:Npn \__cmd_cant_copy:nw #1 #2 \__cmd_break_point:n #3
1006   { \msg_error:nnnn { cmd } { copy-bug } {#1} {#3} }
1007 \msg_new:nnn { cmd } { copy-bug }
```

```

1008 {
1009   Error~while~copying~command~\iow_char:N\\#2:\\
1010   \str_case:nn {#1}
1011   {
1012     { non-ltcmd } { Command~is~not~a~valid~ltcmd~command. }
1013     { unknown-type } { Found~an~unknown~argument~type. }
1014     { invalid-end }
1015     { Target~command~is~not~named~\iow_char:N \\end<name>. }
1016   }
1017 }

```

And, of course, add `_kernel_cmd_if_xparse:NTF` and `_cmd_copy:NN` to `\@declarecommandcopylisthook`:

```

1018 \tl_gput_right:Nn \@declarecommandcopylisthook
1019   { { \_kernel_cmd_if_xparse:NTF \_cmd_copy:NN } }

```

(End definition for `_cmd_copy:NN`, `_cmd_set_eq_if_exist:NN`, and `_cmd_cant_copy:nwn`.)

`_cmd_copy_command:nnNN` `_cmd_copy_command:NnNNnnnn`

A normal (non-expandable) command has a pretty straightforward structure. Its definition is stored in `\⟨cmd⟩_code`, its defaults (if any) are stored in `\⟨cmd⟩_defaults`, and its top-level definition contains its signature, which can just be copied over. `_cmd_copy_command:nnNN` copies the command code and defaults, and then defines the top-level command using the auxiliary `_cmd_copy_command:NnNNnnnn`. This macro takes the signature of the command being copied from its top-level definition, and replaces the named bits with the new name.

```

1020 \cs_new_protected:Npn \_cmd_copy_command:nnNN #1 #2 #3 #4
1021   {
1022     \cs_set_eq:cc { #1 ~ code } { #2 ~ code }
1023     \_cmd_set_eq_if_exist:cc { #1 ~ defaults } { #2 ~ defaults }
1024     \cs_set_protected_nopar:Npx #3
1025     { \exp_after:wN \_cmd_copy_command:NnNNnnnn #4 {#1} }
1026   }
1027 \cs_new:Npn \_cmd_copy_command:NnNNnnnn #1 #2 #3 #4 #5 #6 #7 #8
1028   {
1029     #1 \exp_not:n { {#2} }
1030     \exp_not:c { #8 ~ } \exp_not:c { #8 ~ code }
1031     \exp_not:n { {#5} {#6} {#7} }
1032   }

```

(End definition for `_cmd_copy_command:nnNN` and `_cmd_copy_command:NnNNnnnn`.)

`_cmd_copy_expandable:nnNN` `_cmd_copy_expandable:NnNNNNnn`

An expandable command is slightly more complicated. Besides the `\⟨cmd⟩_code`, and `\⟨cmd⟩_defaults`, it also has an auxiliary `\⟨cmd⟩_l` for grabbing delimited arguments, and possibly another auxiliary `\⟨cmd⟩_ll`, if the command has both long and short arguments. Then, its signature also has several specific bits that are unique to that command; this is in contrast to non-expandable commands, which use a common set of parsing functions.

We start by copying the basics, then call `_cmd_copy_expandable_signature:NnNNNNnnnn` to parse the signature of the command and build up the modified copy in a temporary token list, then we call `_cmd_copy_expandable:NnNNNNnnn` that will copy the top-level definition of the command, with the proper internal renames.

```

1033 \cs_new_protected:Npn \_cmd_copy_expandable:nnNN #1 #2 #3 #4
1034   {
1035     \cs_set_eq:cc { #1 ~ code } { #2 ~ code }

```

```

1036   \__cmd_set_eq_if_exist:cc { #1 ~ } { #2 ~ }
1037   \__cmd_set_eq_if_exist:cc { #1 ~ \c_space_t1 } { #2 ~ \c_space_t1 }
1038   \__cmd_set_eq_if_exist:cc { #1 ~ defaults } { #2 ~ defaults }
1039   \exp_after:wN \__cmd_copy_expandable_signature:NnNNNNnnn #4 {#1} {#2}
1040   \cs_set_nopar:Npx #3
1041     { \exp_after:wN \__cmd_copy_expandable:NnNNNNnnn #4 {#1} {#2} }
1042   }
1043 \cs_new:Npn \__cmd_copy_expandable:NnNNNNnnn #1 #2 #3 #4 #5 #6 #7 #8 #9
1044   {
1045     \exp_not:N #1 \exp_not:n { {#2} }
1046     \exp_not:c { #8 ~ }
1047     \exp_not:c
1048     {
1049       #8 ~
1050       \str_if_eq:eeT
1051         { \exp_not:c { #9 ~ \c_space_t1 } } { \exp_not:N #4 }
1052         { \c_space_t1 }
1053     }
1054     \exp_not:c { #8 ~ code }
1055     \str_if_eq:eeTF { \exp_not:N #6 } { ? }
1056     { ? }
1057     { \exp_not:c { #8 ~ defaults } }
1058     { \exp_not:V \l__cmd_tmpa_t1 }
1059   }

```

A signature for an expandable command contains as many `\expandable_grab_<type>:w` as there are arguments, and what follows this macro depends on the `<type>`. We'll start a loop through the signature, and at each argument grabber, we'll step the argument count, and look for the `<type>` with `__cmd_copy_parse_grabber:w` so that we know which `__cmd_copy_grabber_<type>:w` to call next.

```

1060 \cs_new_protected:Npn \__cmd_copy_expandable_signature:NnNNNNnnn
1061   #1 #2 #3 #4 #5 #6 #7 #8 #9
1062   {
1063     \int_zero:N \l__cmd_current_arg_int
1064     \tl_clear:N \l__cmd_tmpa_t1
1065     \__cmd_copy_expandable:nnN {#8} {#9} #7
1066     \q_recursion_tail \q_recursion_stop
1067   }
1068 \cs_new_protected:Npn \__cmd_copy_expandable:nnN #1 #2 #3
1069   {
1070     \quark_if_recursion_tail_stop:n {#3}
1071     \int_incr:N \l__cmd_current_arg_int
1072     \exp_after:wN \__cmd_copy_parse_grabber:w \token_to_str:N #3 {#1} {#2}
1073   }
1074 \use:x
1075   {
1076     \cs_new_protected:Npn \exp_not:N \__cmd_copy_parse_grabber:w ##1
1077       \tl_to_str:n { expandable_grab_ } ##2 \tl_to_str:n { :w }
1078     {
1079       \tl_put_right:Nx \exp_not:N \l__cmd_tmpa_t1
1080         { \exp_not:N \exp_not:c { __cmd_expandable_grab_##2:w } }
1081         \exp_not:N \cs_if_exist_use:cF { __cmd_copy_grabber_##2:w }
1082           { \__cmd_cant_copy:nwn { unknown-type } }
1083     }

```

```
1084 }
```

The most complicated is the Delimited argument: each argument has a dedicated grabbing function named after the command that has to be copied over (of the form $\langle cmd \rangle \cup (\arg \cup num)$).

```
1085 \cs_new_protected:Npn \__cmd_copy_grabber_D:w #1 #2 #3 #4 #5
1086 {
1087     \tl_put_right:Nx \l__cmd_tmpa_tl
1088     {
1089         \exp_not:c { #1 ~ (arg ~ \int_use:N \l__cmd_current_arg_int ) }
1090         \exp_not:n { #4 #5 }
1091     }
1092     \cs_set_eq:cc
1093         { #1 ~ (arg ~ \int_use:N \l__cmd_current_arg_int ) }
1094         { #2 ~ (arg ~ \int_use:N \l__cmd_current_arg_int ) }
1095     \__cmd_copy_expandable:nnN {#1} {#2}
1096 }
```

D_alt is just a special case of D that uses a single delimiter (used when both delimiters of the argument are identical):

```
1097 \cs_new_protected:Npn \__cmd_copy_grabber_D_alt:w #1 #2 #3 #4
1098     { \__cmd_copy_grabber_D:w {#1} {#2} {#3} {#4} { } }
```

As far as copying is concerned, R is identical to D:

```
1099 \cs_new_eq:NN \__cmd_copy_grabber_R:w \__cmd_copy_grabber_D:w
1100 \cs_new_eq:NN \__cmd_copy_grabber_R_alt:w \__cmd_copy_grabber_D_alt:w
```

E is straightforward: we just copy the embellishments over, and increase the current argument number $\l__cmd_current_arg_int$ by the number of embellishments (minus one because there is a $\int_incr:N$ down the line).

```
1101 \cs_new_protected:Npn \__cmd_copy_grabber_E:w #1 #2 #3 #4
1102 {
1103     \tl_put_right:Nn \l__cmd_tmpa_tl { {#3} {#4} }
1104     \int_add:Nn \l__cmd_current_arg_int { \tl_count:n {#4} - 1 }
1105     \__cmd_copy_expandable:nnN {#1} {#2}
1106 }
1107 \cs_new_eq:NN \__cmd_copy_grabber_E_long:w \__cmd_copy_grabber_E:w
```

t just needs copying the token to be tested for:

```
1108 \cs_new_protected:Npn \__cmd_copy_grabber_t:w #1 #2 #3 #4
1109 {
1110     \tl_put_right:Nn \l__cmd_tmpa_tl { #3 #4 }
1111     \__cmd_copy_expandable:nnN {#1} {#2}
1112 }
```

And last but not least, m is the simplest; the grabber is just $__cmd_expandable_grab_m:w$, which is already added to the new command so here we just resume the loop:

```
1113 \cs_new_protected:Npn \__cmd_copy_grabber_m:w { \__cmd_copy_expandable:nnN }
1114 \cs_new_eq:NN \__cmd_copy_grabber_m_long:w \__cmd_copy_grabber_m:w
```

(End definition for $__cmd_copy_expandable:nnNN$ and others.)

```
\__cmd_copy_environment:nnNN
\__cmd_copy_environment:Nnnnnnn
```

Copying an environment's \begin part is pretty much like copying a command, except it has a longer name, and at the end we have to copy $\environment \langle name \rangle$ into $\langle name \rangle$.

```
1115 \cs_new_protected:Npn \__cmd_copy_environment:nnNN #1 #2 #3 #4
```

```

1116   {
1117     \cs_set_eq:cc { environment~ #1 ~ code } { environment~ #2 ~ code }
1118     \__cmd_set_eq_if_exist:cc
1119       { environment~ #1 ~ defaults } { environment~ #2 ~ defaults }
1120     \cs_set_protected_nopar:cpx { environment~ #1 }
1121       { \exp_after:wN \__cmd_copy_environment:Nnnnnnnn #4 {#1} }
1122     \cs_set_eq:cc {#1} { environment~ #1 }
1123   }
1124 \cs_new:Npn \__cmd_copy_environment:Nnnnnnnn #1 #2 #3 #4 #5 #6 #7
1125   { #1 \exp_not:n { {#2} } {#7} \exp_not:n { {#4} {#5} {#6} } }

```

(End definition for `__cmd_copy_environment:nnNN` and `__cmd_copy_environment:Nnnnnnnn`.)

`__cmd_copy_environment_end:nnNN`
`__cmd_copy_environment_end_aux:nnNN`

Copying an environment's `\end` part is a bit trickier. We first have to make sure that both parts are named `\end<name>` (that's actually not a hard requirement, but an environment `\end` command makes no sense without the `end` in its name), and strip the leading `end` from the strings. After that, copying is straightforward.

```

1126 \cs_new_protected:Npn \__cmd_copy_environment_end:nnNN #1 #2
1127   {
1128     \__cmd_check_end:Nn \l__cmd_tmpa_tl {#1}
1129     \__cmd_check_end:Nn \l__cmd_tmpb_tl {#2}
1130     \exp_args:Nno \__cmd_copy_environment_end_aux:nnNN
1131       { \l__cmd_tmpa_tl } { \l__cmd_tmpb_tl }
1132   }
1133 \cs_new_protected:Npn \__cmd_copy_environment_end_aux:nnNN #1 #2 #3 #4
1134   {
1135     \cs_set_nopar:cp { environment~ #1 ~end }
1136       { \exp_not:c { environment~ #1 ~end~aux } }
1137     \cs_set_eq:cc
1138       { environment~ #1 ~end~aux~ } { environment~ #2 ~end~aux~ }
1139     \cs_set_eq:cc { end #1 } { environment~ #1 ~end }
1140   }

```

To check whether an `\end` command is valid, we look for the string `end` at the beginning of the command name, and if not found, raise an error:

```

\__cmd_check_end:Nn
\__cmd_check_end:n
\__cmd_check_end:w
1141 \cs_new_protected:Npn \__cmd_check_end:Nn #1 #2
1142   {
1143     \tl_set:Nx #1 { \__cmd_check_end:n {#2} }
1144     \token_if_eq_meaning:NNT #1 \q_nil
1145       { \__cmd_cant_copy:nwn { invalid-end } }
1146   }
1147 \cs_set_protected:Npn \__cmd_tmp:w #1
1148   {
1149     \cs_new:Npn \__cmd_check_end:n ##1
1150       {
1151         \exp_after:wN \__cmd_check_end:w \tl_to_str:n {##1}
1152           #1 \q_mark #1 \q_stop
1153       }
1154     \cs_new:Npn \__cmd_check_end:w ##1 #1 ##2 #1 ##3 \q_stop
1155       { \if_meaning:w ##2 \q_mark \exp_not:N \q_nil \else: ##2 \fi: }
1156   }
1157 \exp_args:No \__cmd_tmp:w { \tl_to_str:n { end } }

```

(End definition for `_cmd_copy_environment_end:nNN` and others.)

Not much to do regarding `\texrelease`: we could remove the entries from `\@declarecommandcopylist` but it doesn't seem worth it.

```
1158 <\texrelease> \EndIncludeInRelease  
1159 <\texrelease> \IncludeInRelease{2020/10/01}{\_cmd_copy:NN}%  
1160 <\texrelease> {Support~\NewCommandCopy~in~\ltcmd}  
1161 <\texrelease> \EndIncludeInRelease
```

1.7.2 Showing the definition of a command

```
1162 <\texrelease> \IncludeInRelease{2021/11/15}{\_cmd_show:N}%  
1163 <\texrelease> {Support~\ShowCommand~in~\ltcmd}
```

To show the definition of a command we need more or less the same building blocks as for copying, except that instead of making a copy, we'll just print stuff to the terminal. This macro just branches to the proper showing command by using `_cmd_cmd_type_cases:NnnnnF`. The showing command takes the command to be shown as argument.

```
1164 \cs_new_protected:Npn \_cmd_show:N #1  
1165 {  
1166     \use:x  
1167     {  
1168         \int_set:Nn \tex_escapechar:D { 92 }  
1169         \exp_not:N \_cmd_cmd_type_cases:NnnnnF \exp_not:N #1  
1170         { \_cmd_show_command:N }  
1171         { \_cmd_show_expandable:N }  
1172         { \_cmd_show_environment:N }  
1173         { \_cmd_show_environment_end:N }  
1174         { \_cmd_cant_copy:nwn { non-ltcmd } }  
1175         \exp_not:N #1  
1176         \exp_not:N \_cmd_break_point:n { \cs_to_str:N #1 }  
1177         \int_set:Nn \tex_escapechar:D { \int_use:N \tex_escapechar:D }  
1178     }  
1179 }
```

(End definition for `_cmd_show:N`.)

These commands just expand the command once to reveal its innards, then pass the type of command, the control sequence, the signature, and the code macro to `_cmd_show_command_aux:nNNn`.

```
1180 \cs_new_protected:Npn \_cmd_show_command:N #1  
1181 { \exp_after:wN \_cmd_show_command:NnNwN #1 \q_\cmd #1 }  
1182 \cs_new_protected:Npn \_cmd_show_command:NnNwN #1 #2 #3 #4 #5 \q_\cmd #6  
1183 { \_cmd_show_command_aux:nNNn { document~command } #6 #4 {#2} }  
1184 \cs_new_protected:Npn \_cmd_show_expandable:N #1  
1185 { \exp_after:wN \_cmd_show_expandable:NnNNNnN #1 #1 }  
1186 \cs_new_protected:Npn \_cmd_show_expandable:NnNNNnN #1 #2 #3 #4 #5 #6 #7 #8  
1187 { \_cmd_show_command_aux:nNNn { expandable~document~command } #8 #5 {#2} }
```

Now just print everything in the required format. The auxiliary `_cmd_split_signature:n` stores a ready-to-print token list in `\l_\cmd_tmpa_tl`, so we ust use that here:

```
1188 \cs_new_protected:Npn \_cmd_show_command_aux:nNNn #1 #2 #3 #4  
1189 {  
1190     \_cmd_split_signature:n {#4}}
```

```

1191     \tl_show:x
1192     {
1193         \token_to_str:N #2 = #1: \iow_newline:
1194         \tl_use:N \l__cmd_tmpa_tl
1195         -> \cs_replacement_spec:N #3
1196     }
1197 }
```

We can reuse most of the above to show an environment, except that we need to ensure that the proper `\environment` ... are passed to `_cmd_show_command_aux:nNNn`. Additionally, when `\ShowCommand\foo` is used (if `\foo` is an environment), we show `\endfoo` as well, and when `\ShowCommand\endfoo` is used, change that to `\ShowCommand\foo` and do the same.

```

1198 \cs_new_protected:Npn \_cmd_show_environment:N #
1199 {
1200     \exp_after:wN \_cmd_show_environment:Nnnw #1 \q_ cmd
1201     \tl_show:x
1202     {
1203         \token_to_str:N \end { \cs_to_str:N #1 } : \iow_newline:
1204         -> \exp_args:Nc \cs_replacement_spec:N
1205         { environment~ \cs_to_str:N #1 ~end~aux~ }
1206     }
1207 }
1208 \cs_new_protected:Npn \_cmd_show_environment:Nnnw #1 #2 #3 #4 \q_ cmd
1209 {
1210     \use:x
1211     {
1212         \_cmd_show_command_aux:nNNn { document~environment }
1213         { \exp_not:N \begin {#3} }
1214         \exp_not:c { environment~ #3 ~ code }
1215         {#2}
1216     }
1217 }
1218 \cs_new_protected:Npn \_cmd_show_environment_end:N #
1219 {
1220     \exp_args:NNx \_cmd_check_end:Nn \l__cmd_tmpa_tl { \cs_to_str:N #1 }
1221     \exp_args:Nc \_cmd_show_environment:N { \l__cmd_tmpa_tl }
1222 }
```

And, of course, add `_kernel_cmd_if_xparse:NTF` and `_cmd_show:N` to `\@showcommandlisthook`

```

1223 \tl_gput_right:Nn \@showcommandlisthook
1224 { { \_kernel_cmd_if_xparse:NTF \_cmd_show:N } }
```

(End definition for `_cmd_show_command:N` and others.)

`_cmd_split_signature:n`

Now we'll try a least-effort adventure into splitting the symbolic user-provided signature for a command into individual parameters for pretty-printing. A counter is used to keep track of the current argument number, and two token lists are used: `\l__cmd_tmpa_tl` holds the final token list to be printed, and `\l__cmd_tmpb_tl` holds just the current item, so that we can make changes to an individual item without having to dissect the whole thing (this is used for e- and E-types).

```

1225 \cs_new_protected:Npn \_cmd_split_signature:n #
1226 {
1227     \int_set:Nn \l__cmd_current_arg_int { 1 }
```

```

1228      \tl_clear:N \l__cmd_tmpa_tl
1229      \tl_clear:N \l__cmd_tmpb_tl
1230      \__cmd_split_signature_loop:Nw #1 \q_recursion_tail \q_recursion_stop
1231  }

```

This is the main chunk of the loop: it starts an item with `__cmd_split_start_item:` (this adds indentation and the argument number to `\l__cmd_tmpb_tl`), then checks if a special token list `\c__cmd_show_type_{type}_tl` exists. If it doesn't, the current argument is a "simple" type which needs no extra processing. Otherwise, call a specific function depending on the value of said token list.

```

1232 \cs_new_protected:Npn \__cmd_split_signature_loop:Nw #1
1233 {
1234   \quark_if_recursion_tail_stop:N #1
1235   \tl_if_empty:NT \l__cmd_tmpb_tl { \__cmd_split_start_item: }
1236   \tl_if_exist:cTF { c__cmd_show_type_#1_tl }
1237   {
1238     \use:c
1239     {
1240       __cmd_show_
1241       \if_case:w \tl_use:c { c__cmd_show_type_#1_tl } \exp_stop_f:
1242       delim \or: delims \or: delims_opt \or: opt \or:
1243       e \or: E \or: prefix \or: processor \fi: :Nw
1244     } #1
1245   }
1246   { \__cmd_split_end_item:n {#1} \__cmd_split_signature_loop:Nw }
1247 }

```

The token lists `\c__cmd_show_type_{type}_tl` exist for nontrivial (for printing) `{types}` that require special parsing (like delimiters or optional arguments). Values from 0 to 7 are assigned to each type:

1. a single delimiter token;
2. two delimiter tokens;
3. two delimiter tokens plus a default value;
4. a default value;
5. a list of embellishments (exclusive for e-type);
6. embellishments plus defaults (exclusive for E-type);
7. simple prefixes;
8. prefixes with arguments (argument processors);

```

1248 \cs_set_protected:Npn \__cmd_tmp:w #1 #2
1249 {
1250   \quark_if_nil:nF {#1}
1251   { \tl_const:cn { c__cmd_show_type_#1_tl } {#2} \__cmd_tmp:w }
1252 }
1253 \__cmd_tmp:w t0 r1 d1 R2 D2 03 e4 E5 +6 !6 >7 \q_nil \q_nil

```

Now, based on each type we know how to act. In most cases it is just a matter of feeding in the grabbed arguments and resuming the loop. The embellishments require a bit more attention: the e-type loops through the list of embellishments and adds each to the token list as a separate argument. The E-type does more or less the same, but uses `__cmd_tl_mapthread_function:nnN` to map over two lists simultaneously, adding each token and default to the token list for printing.

```

1254 \cs_new_protected:Npn \__cmd_show_delim:Nw #1 #2
1255   { \__cmd_split_end_item:n { #1 #2 } \__cmd_split_signature_loop:Nw }
1256 \cs_new_protected:Npn \__cmd_show_delims:Nw #1 #2 #3
1257   { \__cmd_split_end_item:n { #1 #2 #3 } \__cmd_split_signature_loop:Nw }
1258 \cs_new_protected:Npn \__cmd_show_delims_opt:Nw #1 #2 #3 #4
1259   { \__cmd_split_end_item:n { #1 #2 #3 {#4} } \__cmd_split_signature_loop:Nw }
1260 \cs_new_protected:Npn \__cmd_show_opt:Nw #1 #2
1261   { \__cmd_split_end_item:n { #1 {#2} } \__cmd_split_signature_loop:Nw }
1262 \cs_new_protected:Npn \__cmd_show_e:Nw #1 #2
1263   {
1264     \tl_map_inline:nn {#2}
1265   {
1266     \__cmd_split_start_item:
1267     \__cmd_split_end_item:n { #1 ##1 }
1268   }
1269   \__cmd_split_signature_loop:Nw
1270 }
1271 \cs_set_protected:Npn \__cmd_tmp:w #1
1272   {
1273     \cs_new_protected:Npn \__cmd_show_E:Nw ##1 ##2 ##3
1274   {
1275     \cs_set_protected:Npn \__cmd_tmp:w #####1 #####2
1276   {
1277     \__cmd_split_start_item:
1278     \__cmd_split_end_item:n { ##1 #####1 #####2 }
1279   }
1280   \__cmd_tl_mapthread_function:nnN {##2}
1281   { ##3 {#1} {#1} {#1} {#1} {#1} {#1} {#1} {#1} } \__cmd_tmp:w
1282   \__cmd_split_signature_loop:Nw
1283 }
1284 }
1285 \exp_args:NV \__cmd_tmp:w \c_novalue_tl

```

Minor wrinkle with the prefixes: they use `__cmd_split_add_item:n` instead of `__cmd_split_end_item:n` (add *vs.* end) because they are followed by an argument, so they can't end the item.

```

1286 \cs_new_protected:Npn \__cmd_show_prefix:Nw #
1287   { \__cmd_split_add_item:n {#1} \__cmd_split_signature_loop:Nw }
1288 \cs_new_protected:Npn \__cmd_show_processor:Nw #1 #2
1289   { \__cmd_split_add_item:n { #1 {#2} } \__cmd_split_signature_loop:Nw }

```

And now the auxiliaries that store the strings to be printed. `__cmd_split_start_item:` starts an item from scratch, `__cmd_split_add_item:n` adds tokens to an item without adding a newline, and `__cmd_split_end_item:n` adds tokens, terminates the item with a newline, and steps the argument count.

```

1290 \cs_new_protected:Npn \__cmd_split_start_item:
1291   {

```

```
1292     \tl_set:Nx \l__cmd_tmpb_tl
1293     { ~ \c_space_tl \c_hash_str \int_use:N \l__cmd_current_arg_int : }
1294 }
1295 \cs_new_protected:Npn \__cmd_split_add_item:n #1
1296 { \tl_put_right:Nx \l__cmd_tmpb_tl { \tl_to_str:n {#1} } }
1297 \cs_new_protected:Npn \__cmd_split_end_item:n #1
1298 {
1299     \tl_put_right:Nx \l__cmd_tmptl
1300     { \l__cmd_tmpb_tl \tl_to_str:n {#1} \iow_newline: }
1301     \tl_clear:N \l__cmd_tmpb_tl
1302     \int_incr:N \l__cmd_current_arg_int
1303 }
```

(End definition for `_cmd_split_signature:n` and others.)

Not much to do regarding `\textrm{latexrelease}`: we could remove the entries from `\@showcommandlisthook`, but it doesn't seem worth it.

```
1304 \end{IncludeInRelease}
1305 %
1306 \IncludeInRelease{2020/10/01}{\_cmd_show:N}%
1307 {\Support~\ShowCommand~in~\ltcmd}
1308 \end{IncludeInRelease}
```

1.8 Grabbing arguments

All of the grabbers follow the same basic pattern. The initial function stores in \lCmdSignatureTl the code to grab further arguments, defines (the function in) \lCmdFnTl that will grab the argument, and calls it.

Defining `\l1__cmd_fn_t1` means determining whether to use `\cs_set:Npn` or `\cs_set_nopar:Npn`, and for optional arguments whether to skip spaces. Once the argument is found, `\l1__cmd_fn_t1` calls `__cmd_add_arg:n`, responsible for calling processors and grabbing further arguments.

```

  \__cmd_grab_b:w
  \__cmd_grab_b_long:w
\__cmd_grab_b_obey_spaces:w
  \__cmd_grab_b_long_obey_spaces:w
  \__cmd_grab_b_aux:NNw
  \__cmd_grab_b_end:Nw

```

This uses the well-tested code of D-type arguments, skipping the peeking step because the b-type argument is always present, and adding a cleanup stage at the end by hijacking the signature. The clean-up consists of properly dealing with `\l__cmd_args_t1` and also putting back the `\end` that served as an end-delimiter: this `\end` receives the environment name as its argument and is run normally. The D-type code stores the argument found (body of the environment) as a brace group in `\l__cmd_args_t1` and depending on the presence of a prefix ! we trim spaces or not before adding this braced argument into the saved `\l__cmd_args_t1`. The strange `\begin_` control sequence is there for display purposes only: it has to look like `\begin` in the terminal but not to delimited arguments.

```
1309 \cs_new_protected:Npn \__cmd_grab_b:w
1310   { \__cmd_grab_b_aux:NNw \cs_set_protected_nopar:Npn \tl_trim_spaces:n }
1311 \cs_new_protected:Npn \__cmd_grab_b_long:w
1312   { \__cmd_grab_b_aux:NNw \cs_set_protected:Npn \tl_trim_spaces:n }
1313 \cs_new_protected:Npn \__cmd_grab_b_obey_spaces:w
1314   { \__cmd_grab_b_aux:NNw \cs_set_protected_nopar:Npn \exp_not:n }
1315 \cs_new_protected:Npn \__cmd_grab_b_long_obey_spaces:w
1316   { \__cmd_grab_b_aux:NNw \cs_set_protected:Npn \exp_not:n }
1317 \cs_new_protected:Npn \__cmd_grab_b_aux:NNw #1#2#3 \__cmd_run_code:
1318   {
1319     \__cmd_grab_D_aux>NNnN \begin \end {#3} #1
```

```

1320      \tl_put_left:Nn \l__cmd_signature_tl { \__cmd_grab_b_end:Nw #2 }
1321      \tl_set_eq:NN \l__cmd_saved_args_tl \l__cmd_args_tl
1322      \tl_clear:N \l__cmd_args_tl
1323      \exp_args:Nc \l__cmd_fn_tl { begin ~ }
1324  }
1325 \cs_new_protected:Npn \__cmd_grab_b_end:Nw #1#2 \__cmd_run_code:
1326 {
1327     \tl_set:Nx \l__cmd_args_tl
1328     {
1329         \exp_not:V \l__cmd_saved_args_tl
1330         { \exp_after:wN #1 \l__cmd_args_tl }
1331     }
1332     #2
1333     \__cmd_run_code:
1334     \end
1335 }

```

(End definition for `__cmd_grab_b:w` and others.)

`__cmd_grab_D:w`
`__cmd_grab_D_long:w`
`__cmd_grab_D_obey_spaces:w`

```

\__cmd_grab_D_long_obey_spaces:w
1336 \cs_new_protected:Npn \__cmd_grab_D:w #1#2#3 \__cmd_run_code:
1337 {
1338     \__cmd_grab_D_aux:NNnNN #1 #2 {#3} \cs_set_protected_nopar:Npn
1339     \__cmd_peek_nonspace_remove:NTF
1340 }
1341 \cs_new_protected:Npn \__cmd_grab_D_long:w #1#2#3 \__cmd_run_code:
1342 {
1343     \__cmd_grab_D_aux:NNnNN #1 #2 {#3} \cs_set_protected:Npn
1344     \__cmd_peek_nonspace_remove:NTF
1345 }
1346 \cs_new_protected:Npn \__cmd_grab_D_obey_spaces:w #1#2#3 \__cmd_run_code:
1347 {
1348     \__cmd_grab_D_aux:NNnNN #1 #2 {#3} \cs_set_protected_nopar:Npn
1349     \__cmd_peek_meaning_remove:NTF
1350 }
1351 \cs_new_protected:Npn \__cmd_grab_D_long_obey_spaces:w #1#2#3 \__cmd_run_code:
1352 {
1353     \__cmd_grab_D_aux:NNnNN #1 #2 {#3} \cs_set_protected:Npn
1354     \__cmd_peek_meaning_remove:NTF
1355 }

```

This is a bit complicated. The idea is that, in order to check for nested optional argument tokens ([[[...]]] and so on) the argument needs to be grabbed without removing any braces at all. If this is not done, then cases like [{[]}] fail. So after testing for an optional argument, it is collected piece-wise. Inserting a quark prevents loss of braces, and there is then a test to see if there are nested delimiters to handle.

```

\__cmd_grab_D_aux:NNnNN
\__cmd_grab_D_aux:NNnN
1356 \cs_new_protected:Npn \__cmd_grab_D_aux:NNnNN #1#2#3#4#5
1357 {
1358     \__cmd_grab_D_aux:NNnN #1#2 {#3} #4
1359     #5 #1
1360     { \__cmd_grab_D_call:Nw #1 }
1361     { \__cmd_add_arg:o \c_novalue_tl }

```

```
1362 }
```

Inside the “standard” grabber, there is a test to see if the grabbed argument is entirely enclosed by braces. There are a couple of extra factors to allow for: the argument might be entirely empty, and spaces at the start and end of the input must be retained around a brace group. Also notice that a *blank* argument might still contain spaces.

```
1363 \cs_new_protected:Npn \__cmd_grab_D_aux:NNnN #1#2#3#4
1364 {
1365   \tl_set:Nn \l__cmd_signature_tl {#3}
1366   \exp_after:wN #4 \l__cmd_fn_tl ##1 #2
1367   {
1368     \tl_if_in:nnTF {##1} {#1}
1369     { \__cmd_grab_D_nested:NNnN #1 #2 {##1} #4 }
1370     {
1371       \tl_if_blank:oTF { \use_none:n ##1 }
1372       { \__cmd_add_arg:o { \use_none:n ##1 } }
1373       {
1374         \str_if_eq:eeTF
1375           { \exp_not:o { \use_none:n ##1 } }
1376           { { \exp_not:o { \use_i:nn ##1 \q_nil } } }
1377           { \__cmd_add_arg:o { \use_i:nn ##1 } }
1378           { \__cmd_add_arg:o { \use_none:n ##1 } }
1379       }
1380     }
1381   }
1382 }
```

(End definition for `__cmd_grab_D:w` and others.)

```
\__cmd_grab_D_nested:NNnN
\__cmd_grab_D_nested:w
\l__cmd_nesting_a_tl
\l__cmd_nesting_b_tl
\q__cmd
```

Catching nested optional arguments means more work. The aim here is to collect up each pair of optional tokens without TeX helping out, and without counting anything. The code above will already have removed the leading opening token and a closing token, but the wrong one. The aim is then to work through the material grabbed so far and divide it up on each opening token, grabbing a closing token to match (thus working in pairs). Once there are no opening tokens, then there is a second check to see if there are any opening tokens in the second part of the argument (for things like `[] []`). Once everything has been found, the entire collected material is added to the output as a single argument. The only tricky part here is ensuring that any grabbing function that might run away is named after the function currently being parsed and not after `xparse`. That leads to some rather complex nesting! There is also a need to prevent the loss of any braces, hence the insertion and removal of quarks along the way.

```
1383 \tl_new:N \l__cmd_nesting_a_tl
1384 \tl_new:N \l__cmd_nesting_b_tl
1385 \quark_new:N \q__cmd
1386 \cs_new_protected:Npn \__cmd_grab_D_nested:NNnN #1#2#3#4
1387 {
1388   \tl_clear:N \l__cmd_nesting_a_tl
1389   \tl_clear:N \l__cmd_nesting_b_tl
1390   \exp_after:wN #4 \l__cmd_fn_tl ##1 #1 ##2 \q__cmd ##3 #2
1391   {
1392     \tl_put_right:No \l__cmd_nesting_a_tl { \use_none:n ##1 #1 }
1393     \tl_put_right:No \l__cmd_nesting_b_tl { \use_i:nn ##2 ##3 }
1394     \tl_if_in:nnTF {##2} {#1}
```

```

1395      {
1396          \l__cmd_fn_tl
1397          \q_nil ##2 \q_cmd \ERROR
1398      }
1399      {
1400          \tl_put_right:Nx \l__cmd_nesting_a_tl
1401          { \__cmd_grab_D_nested:w \q_nil ##2 \q_stop }
1402          \tl_if_in:NnTF \l__cmd_nesting_b_tl {#1}
1403          {
1404              \tl_set_eq:NN \l__cmd_tmpa_tl \l__cmd_nesting_b_tl
1405              \tl_clear:N \l__cmd_nesting_b_tl
1406              \exp_after:wN \l__cmd_fn_tl \exp_after:wN
1407                  \q_nil \l__cmd_tmpa_tl \q_nil \q_cmd \ERROR
1408          }
1409          {
1410              \tl_put_right:No \l__cmd_nesting_a_tl
1411                  \l__cmd_nesting_b_tl
1412                  \__cmd_add_arg:V \l__cmd_nesting_a_tl
1413          }
1414      }
1415  }
1416  \l__cmd_fn_tl #3 \q_nil \q_cmd \ERROR
1417 }
1418 \cs_new:Npn \__cmd_grab_D_nested:w #1 \q_nil \q_stop
1419  { \exp_not:o { \use_none:n #1 } }

(End definition for \__cmd_grab_D_nested:NNnN and others.)

```

__cmd_grab_D_call:Nw For D and R-type arguments, to avoid losing any braces, a token needs to be inserted before the argument to be grabbed. If the argument runs away because the closing token is missing then this inserted token shows up in the terminal. Ideally, #1 would therefore be used directly, but that is no good as it will mess up the rest of the grabber. Instead, a copy of #1 with an altered category code is used, as this will look right in the terminal but will not mess up the grabber. The only issue then is that the category code of #1 is unknown. So there is a quick test to ensure that the inserted token can never be matched by the grabber. (This assumes that the open and close delimiters are not the same character with different category codes, but that really should not happen in any sensible document-level syntax.) An exception is when #1 is a control sequence token, in which case the character-token treatment is no good because if hit with \token_to_str:N it would add sputios tokens to the argument. In this case a different branch is taken. The token inserted is then the same <csname> as #1, but with a space appended, so that the grabber don't see it as another of the same delimiter.

```

1420 \cs_new_protected_nopar:Npn \__cmd_grab_D_call:Nw #1
1421  {
1422      \token_if_eq_catcode:NNTF + #1
1423      {
1424          \exp_after:wN \exp_after:wN \exp_after:wN
1425              \l__cmd_fn_tl \char_generate:nn { '#1 } { 11 }
1426      }
1427      {
1428          \__cmd_token_if_cs:NTF #1
1429          {
1430              \exp_after:wN \l__cmd_fn_tl

```

```

1431           \cs:w \cs_to_str:N #1 ~ \cs_end:
1432       }
1433   {
1434     \exp_after:wN \l__cmd_fn_tl
1435     \token_to_str:N #1
1436   }
1437 }
1438 }
```

(End definition for `_cmd_grab_D_call:Nw`.)

`_cmd_grab_E:w` Everything here needs to point to a loop.

```

\__cmd_grab_E_long:w 1439 \cs_new_protected:Npn \__cmd_grab_E:w #1#2 \__cmd_run_code:
\__cmd_grab_E_obey_spaces:w 1440 {
  \__cmd_grab_E_long_obey_spaces:w 1441   \__cmd_grab_E:nnNN {#1} {#2}
  \__cmd_grab_E:nnNN 1442   \cs_set_protected_nopar:Npn
  \__cmd_grab_E_loop:NnN 1443   \__cmd_peek_nonspace_remove:NTF
\__cmd_grab_E_finalise: 1444 }
```

```

1445 \cs_new_protected:Npn \__cmd_grab_E_long:w #1#2 \__cmd_run_code:
1446 {
  \__cmd_grab_E:nnNN {#1} {#2}
  \cs_set_protected:Npn
  \__cmd_peek_nonspace_remove:NTF
1449 }
```

```

1450 \cs_new_protected:Npn \__cmd_grab_E_obey_spaces:w #1#2 \__cmd_run_code:
1451 {
  \__cmd_grab_E:nnNN {#1} {#2}
  \cs_set_protected_nopar:Npn
  \__cmd_peek_meaning_remove:NTF
1455 }
```

```

1456 \cs_new_protected:Npn \__cmd_grab_E_long_obey_spaces:w #1#2 \__cmd_run_code:
1457 {
  \__cmd_grab_E:nnNN {#1} {#2}
  \cs_set_protected:Npn
  \__cmd_peek_meaning_remove:NTF
1461 }
```

A loop is needed here to allow a random ordering of keys. These are searched for one at a time, with any not found needing to be tracked: they can appear later. The grabbed values are held in a property list which is then turned into an ordered list to be passed back to the user.

```

1463 \cs_new_protected:Npn \__cmd_grab_E:nnNN #1#2#3#4
1464 {
1465   \exp_after:wN #3 \l__cmd_fn_tl ##1##2##3
1466   {
1467     \prop_put:Nnn \l__cmd_tmp_prop {##1} {##3}
1468     \__cmd_grab_E_loop:NnN #4 { } ##2 \q_recursion_stop
1469   }
1470   \prop_clear:N \l__cmd_tmp_prop
1471   \tl_set:Nn \l__cmd_signature_tl {#2}
1472   \cs_set_protected:Npn \__cmd_grab_E_finalise:
1473   {
1474     \tl_map_inline:nn {#1}
1475   }
```

```

1476          \prop_get:NnNF \l__cmd_tmp_prop {####1} \l__cmd_tmpb_tl
1477          { \tl_set_eq:NN \l__cmd_tmpb_tl \c_novalue_tl }
1478          \tl_put_right:Nx \l__cmd_args_tl
1479          { { \exp_not:V \l__cmd_tmpb_tl } }
1480      }
1481      \l__cmd_signature_tl \__cmd_run_code:
1482  }
1483  \__cmd_grab_E_loop:NnN #4 { } #1 \q_recursion_tail \q_recursion_stop
1484 }
1485 \cs_new_protected:Npn \__cmd_grab_E_loop:NnN #1#2#3#4 \q_recursion_stop
1486 {
1487     \cs_if_eq:NNTF #3 \q_recursion_tail
1488     { \__cmd_grab_E_finalise: }
1489     {
1490         #1 #3
1491         { \l__cmd_fn_tl #3 {#2#4} }
1492         { \__cmd_grab_E_loop:NnN #1 {#2#3} #4 \q_recursion_stop }
1493     }
1494 }
1495 \cs_new_protected:Npn \__cmd_grab_E_finalise: { }

(End definition for \__cmd_grab_E:w and others.)

```

__cmd_grab_m:w Collecting a single mandatory argument is quite easy.

```

\__cmd_grab_m_long:w 1496 \cs_new_protected:Npn \__cmd_grab_m:w #1 \__cmd_run_code:
1497 {
1498     \tl_set:Nn \l__cmd_signature_tl {#1}
1499     \exp_after:wN \cs_set_protected_nopar:Npn \l__cmd_fn_tl ##1
1500     { \__cmd_add_arg:n {##1} }
1501     \l__cmd_fn_tl
1502 }
1503 \cs_new_protected:Npn \__cmd_grab_m_long:w #1 \__cmd_run_code:
1504 {
1505     \tl_set:Nn \l__cmd_signature_tl {#1}
1506     \exp_after:wN \cs_set_protected:Npn \l__cmd_fn_tl ##1
1507     { \__cmd_add_arg:n {##1} }
1508     \l__cmd_fn_tl
1509 }

```

(End definition for __cmd_grab_m:w and __cmd_grab_m_long:w.)

__cmd_grab_m_1:w - __cmd_grab_m_9:w Grabbing 1–8 mandatory arguments is done by giving 8–1 known arguments to a 9-argument function that stores them in \l__cmd_args_tl. For simplicity, grabbing 9 mandatory arguments is done by grabbing 5 then 4 arguments.

```

\__cmd_grab_m_4:w 1510 \cs_new_protected_nopar:Npn \__cmd_grab_m_aux:Nnnnnnnnn #1#2#3#4#5#6#7#8#9
\__cmd_grab_m_5:w 1511 {
\__cmd_grab_m_6:w 1512     \tl_put_right:No \l__cmd_args_tl
\__cmd_grab_m_7:w 1513     { #1 {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9} }
\__cmd_grab_m_8:w 1514     \l__cmd_signature_tl \__cmd_run_code:
\__cmd_grab_m_9:w 1515 }
\__cmd_grab_m_aux:Nnnnnnnnn 1516 \cs_new_protected:cpn { __cmd_grab_m_1:w } #1 \__cmd_run_code:
1517 {
1518     \tl_set:Nn \l__cmd_signature_tl {#1}
1519     \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn

```

```

1520      \l__cmd_fn_tl \use_none:nnnnnnn { } { } { } { } { } { } { }
1521    }
1522 \cs_new_protected:cpn { __cmd_grab_m_2:w } #1 \__cmd_run_code:
1523  {
1524    \tl_set:Nn \l__cmd_signature_tl {#1}
1525    \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
1526    \l__cmd_fn_tl \use_none:nnnnnn { } { } { } { } { } { }
1527  }
1528 \cs_new_protected:cpn { __cmd_grab_m_3:w } #1 \__cmd_run_code:
1529  {
1530    \tl_set:Nn \l__cmd_signature_tl {#1}
1531    \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
1532    \l__cmd_fn_tl \use_none:nnnn { } { } { } { } { }
1533  }
1534 \cs_new_protected:cpn { __cmd_grab_m_4:w } #1 \__cmd_run_code:
1535  {
1536    \tl_set:Nn \l__cmd_signature_tl {#1}
1537    \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
1538    \l__cmd_fn_tl \use_none:nnnn { } { } { } { }
1539  }
1540 \cs_new_protected:cpn { __cmd_grab_m_5:w } #1 \__cmd_run_code:
1541  {
1542    \tl_set:Nn \l__cmd_signature_tl {#1}
1543    \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
1544    \l__cmd_fn_tl \use_none:nnn { } { } { }
1545  }
1546 \cs_new_protected:cpn { __cmd_grab_m_6:w } #1 \__cmd_run_code:
1547  {
1548    \tl_set:Nn \l__cmd_signature_tl {#1}
1549    \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
1550    \l__cmd_fn_tl \use_none:nn { } { }
1551  }
1552 \cs_new_protected:cpn { __cmd_grab_m_7:w } #1 \__cmd_run_code:
1553  {
1554    \tl_set:Nn \l__cmd_signature_tl {#1}
1555    \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
1556    \l__cmd_fn_tl \use_none:n { }
1557  }
1558 \cs_new_protected:cpn { __cmd_grab_m_8:w } #1 \__cmd_run_code:
1559  {
1560    \tl_set:Nn \l__cmd_signature_tl {#1}
1561    \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
1562    \l__cmd_fn_tl \prg_do_nothing:
1563  }
1564 \cs_new_protected:cpx { __cmd_grab_m_9:w }
1565  {
1566    \exp_not:c { __cmd_grab_m_5:w }
1567    \exp_not:c { __cmd_grab_m_4:w }
1568  }

```

(End definition for `__cmd_grab_m_1:w` and others.)

`__cmd_grab_R:w` The grabber for R-type arguments is basically the same as that for D-type ones, but
`__cmd_grab_R_long:w` always skips spaces (as it is mandatory) and has a hard-coded error message.
`__cmd_grab_R_aux:NNnN`

```

1569 \cs_new_protected:Npn \__cmd_grab_R:w #1#2#3 \__cmd_run_code:
1570   { \__cmd_grab_R_aux:NNnN #1 #2 {#3} \cs_set_protected_nopar:Npn }
1571 \cs_new_protected:Npn \__cmd_grab_R_long:w #1#2#3 \__cmd_run_code:
1572   { \__cmd_grab_R_aux:NNnN #1 #2 {#3} \cs_set_protected:Npn }
1573 \cs_new_protected:Npn \__cmd_grab_R_aux:NNnN #1#2#3#4
1574   {
1575     \__cmd_grab_D_aux:NNnN #1 #2 {#3} #4
1576     \__cmd_peek_nonspace_remove:NTF #1
1577     { \__cmd_grab_D_call:Nw #1 }
1578     {
1579       \msg_error:nxxx { cmd } { missing-required }
1580       { \__cmd_environment_or_command: }
1581       { \token_to_str:N #1 }
1582       \__cmd_add_arg:o \c_novalue_tl
1583     }
1584   }

```

(End definition for `__cmd_grab_R:w`, `__cmd_grab_R_long:w`, and `__cmd_grab_R_aux:NNnN`.)

Dealing with a token is quite easy. Check the match, remove the token if needed and add a flag to the output.

```

1585 \cs_new_protected:Npn \__cmd_grab_t:w
1586   { \__cmd_grab_t_aux:NNw \__cmd_peek_nonspace_remove:NTF }
1587 \cs_new_protected:Npn \__cmd_grab_t_obey_spaces:w
1588   { \__cmd_grab_t_aux:NNw \__cmd_peek_meaning_remove:NTF }
1589 \cs_new_protected:Npn \__cmd_grab_t_aux:NNw #1#2#3 \__cmd_run_code:
1590   {
1591     \tl_set:Nn \l__cmd_signature_tl {#3}
1592     \exp_after:wN \cs_set_protected:Npn \l__cmd_fn_tl
1593     {
1594       #1 #2
1595       { \__cmd_add_arg:n { \BooleanTrue } }
1596       { \__cmd_add_arg:n { \BooleanFalse } }
1597     }
1598     \l__cmd_fn_tl
1599   }

```

(End definition for `__cmd_grab_t:w`, `__cmd_grab_t_obey_spaces:w`, and `__cmd_grab_t_aux:NNw`.)

`\l__cmd_v_arg_tl`

The opening delimiter is the first non-space token, and is never read verbatim. This is required by consistency with the case where the preceding argument was optional and absent: then TeX has already read and tokenized that token when looking for the optional argument. The first thing is thus to check is that this delimiter is a character, and to distinguish the case of a left brace (in that case, `\group_align_safe_end:` is needed to compensate for the begin-group character that was just seen). Then set verbatim catcodes with `__cmd_grab_v_aux_catcodes:`.

The group keep catcode changes local, and `\group_align_safe_begin/end:` allow to use a character with category code 4 (normally `&`) as the delimiter (all commands do `\group_align_safe_begin/end:`, so there's no need to do that again here). It is

ended by `__cmd_grab_v_group_end`, which smuggles the collected argument out of the group.

```

1601 \cs_new_protected:Npn \__cmd_grab_v:w
1602   {
1603     \bool_set_false:N \l__cmd_long_bool
1604     \__cmd_grab_v_aux:w
1605   }
1606 \cs_new_protected:Npn \__cmd_grab_v_long:w
1607   {
1608     \bool_set_true:N \l__cmd_long_bool
1609     \__cmd_grab_v_aux:w
1610   }
1611 \cs_new_protected:Npn \__cmd_grab_v_aux:w #1 \__cmd_run_code:
1612   {
1613     \tl_set:Nn \l__cmd_signature_tl {#1}
1614     \group_begin:
1615       \tex_escapechar:D = 92 \scan_stop:
1616       \tl_clear:N \l__cmd_v_arg_tl
1617       \peek_remove_spaces:n
1618       {
1619         \peek_meaning_remove:NTF \c_group_begin_token
1620         {
1621           \group_align_safe_end:
1622           \__cmd_grab_v_bgroup:
1623         }
1624       {
1625         \peek_N_type:TF
1626           { \__cmd_grab_v_aux_test:N }
1627           { \__cmd_grab_v_aux_abort:n { } }
1628       }
1629     }
1630   }
1631 \cs_new_protected:Npn \__cmd_grab_v_group_end:
1632   {
1633     \exp_args:NNN
1634     \group_end:
1635     \tl_set:Nn \l__cmd_v_arg_tl { \l__cmd_v_arg_tl }
1636   }

```

(End definition for `__cmd_grab_v:w` and others.)

`__cmd_grab_v_aux_test:N` Check that the opening delimiter is a character, setup category codes, then start reading tokens one by one, keeping the delimiter as an argument. If the verbatim was not nested, we will be grabbing one character at each step. Unfortunately, it can happen that what follows the verbatim argument is already tokenized. Thus, we check at each step that the next token is indeed a “nice” character, *i.e.*, is not a character with category code 1 (begin-group), 2 (end-group) or 6 (macro parameter), nor the space character, with category code 10 and character code 32, nor a control sequence. The partially built argument is stored in `\l__cmd_v_arg_tl`. If we ever meet a token which we cannot grab (non-N-type), or which is not a character according to `__cmd_grab_v_token_if_char:NTF`, then we bail out with `__cmd_grab_v_aux_abort:n`. Otherwise, we stop at the first character matching the delimiter.

```
1637 \cs_new_protected:Npn \__cmd_grab_v_aux_test:N #1
```

```

1638   {
1639     \__cmd_grab_v_token_if_char:NTF #1
1640     {
1641       \__cmd_grab_v_aux_put:N #1
1642       \__cmd_grab_v_aux_catcodes:
1643       \__cmd_grab_v_aux_loop:N #1
1644     }
1645     { \__cmd_grab_v_aux_abort:n {#1} #1 }
1646   }
1647 \cs_new_protected:Npn \__cmd_grab_v_aux_loop:N #1
1648   {
1649     \peek_N_type:TF
1650     { \__cmd_grab_v_aux_loop:NN #1 }
1651     { \__cmd_grab_v_aux_abort:n { } }
1652   }
1653 \cs_new_protected:Npn \__cmd_grab_v_aux_loop:NN #1#2
1654   {
1655     \__cmd_grab_v_token_if_char:NTF #2
1656     {
1657       \token_if_eq_charcode:NNTF #1 #2
1658       { \__cmd_grab_v_aux_loop_end: }
1659       {
1660         \__cmd_grab_v_aux_put:N #2
1661         \__cmd_grab_v_aux_loop:N #1
1662       }
1663     }
1664     { \__cmd_grab_v_aux_abort:n {#2} #2 }
1665   }
1666 \cs_new_protected:Npn \__cmd_grab_v_aux_loop_end:
1667   {
1668     \__cmd_grab_v_group_end:
1669     \__cmd_add_arg:x { \tl_tail:N \l__cmd_v_arg_tl }
1670   }

```

(End definition for `__cmd_grab_v_aux_test:N` and others.)

`\l__cmd_v_nesting_int` 1671 `\int_new:N \l__cmd_v_nesting_int`

`__cmd_grab_v_bgroup:` If the opening delimiter is a left brace, we keep track of how many left and right braces were encountered so far in `\l__cmd_v_nesting_int` (the methods used for optional arguments cannot apply here), and stop as soon as it reaches 0.

Some care was needed when removing the opening delimiter, which has already been assigned category code 1: using `\peek_meaning_remove:NTF` in the `__cmd_grab_v_aux:w` function would break within alignments. Instead, we first convert that token to a string, and remove the result as a normal undelimited argument.

```

1672 \cs_new_protected:Npx \__cmd_grab_v_bgroup:
1673   {
1674     \exp_not:N \__cmd_grab_v_aux_catcodes:
1675     \exp_not:n { \int_set:Nn \l__cmd_v_nesting_int { 1 } }
1676     \exp_not:N \__cmd_grab_v_aux_put:N \iow_char:N \{
1677     \exp_not:N \__cmd_grab_v_bgroup_loop:
1678   }

```

```

1679 \cs_new_protected:Npn \__cmd_grab_v_bgroup_loop:
1680 {
1681   \peek_N_type:TF
1682   { \__cmd_grab_v_bgroup_loop:N }
1683   { \__cmd_grab_v_aux_abort:n {} }
1684 }
1685 \cs_new_protected:Npn \__cmd_grab_v_bgroup_loop:N #1
1686 {
1687   \__cmd_grab_v_token_if_char:NTF #1
1688   {
1689     \token_if_eq_charcode:NNTF \c_group_end_token #1
1690     {
1691       \int_decr:N \l__cmd_v_nesting_int
1692       \int_compare:nNnTF \l__cmd_v_nesting_int > 0
1693       {
1694         \__cmd_grab_v_aux_put:N #1
1695         \__cmd_grab_v_bgroup_loop:
1696       }
1697       { \__cmd_grab_v_aux_loop_end: }
1698     }
1699     {
1700       \token_if_eq_charcode:NNT \c_group_begin_token #1
1701       { \int_incr:N \l__cmd_v_nesting_int }
1702       \__cmd_grab_v_aux_put:N #1
1703       \__cmd_grab_v_bgroup_loop:
1704     }
1705   }
1706   { \__cmd_grab_v_aux_abort:n {#1} #1 }
1707 }

```

(End definition for `__cmd_grab_v_bgroup:`, `__cmd_grab_v_bgroup_loop:`, and `__cmd_grab_v_bgroup_loop:N`.)

`__cmd_grab_v_aux_catcodes:` The approach for short verbatim arguments is to make the end-line character a macro parameter character: this is forbidden by the rest of the code. Then the error branch can check what caused the bail out and give the appropriate error message.

```

1708 \cs_new_protected:Npn \__cmd_grab_v_aux_catcodes:
1709 {
1710   \cs_set_eq:NN \do \char_set_catcode_other:N
1711   \dospecials
1712   \tex_endlinechar:D = ‘\^M \scan_stop:
1713   \bool_if:NTF \l__cmd_long_bool
1714   { \char_set_catcode_other:n { \tex_endlinechar:D } }
1715   { \char_set_catcode_parameter:n { \tex_endlinechar:D } }
1716 }
1717 \cs_new_protected:Npn \__cmd_grab_v_aux_abort:n #1
1718 {
1719   \__cmd_grab_v_group_end:
1720   \exp_after:wN \exp_after:wN \exp_after:wN
1721   \peek_meaning_remove:NTF \char_generate:nn { \tex_endlinechar:D } { 6 }
1722   {
1723     \msg_error:nnxxxx { cmd } { verbatim-nl }
1724     { \__cmd_environment_or_command: }
1725     { \tl_to_str:N \l__cmd_v_arg_t1 }

```

```

1726         { \tl_to_str:n {#1} }
1727         \__cmd_add_arg:o \c_novalue_tl
1728     }
1729     {
1730         \msg_error:nnn { cmd } { verbatim-tokenized }
1731         { \__cmd_environment_or_command: }
1732         { \tl_to_str:N \l__cmd_v_arg_tl }
1733         { \tl_to_str:n {#1} }
1734         \__cmd_add_arg:o \c_novalue_tl
1735     }
1736 }

```

(End definition for `__cmd_grab_v_aux_catcodes:` and `__cmd_grab_v_aux_abort:n.`)

`__cmd_grab_v_aux_put:N` Storing one token in the collected argument. Most tokens are converted to category code 12, with the exception of active characters, and spaces (not sure what should be done for those).

```

1737 \cs_new_protected:Npn \__cmd_grab_v_aux_put:N #1
1738 {
1739     \tl_put_right:Nx \l__cmd_v_arg_tl
1740     {
1741         \token_if_active:NTF #1
1742         { \exp_not:N #1 } { \token_to_str:N #1 }
1743     }
1744 }

```

(End definition for `__cmd_grab_v_aux_put:N.`)

`__cmd_grab_v_token_if_char:NTF` This function assumes that the escape character is printable. Then the string representation of control sequences is at least two characters, and `\str_tail:n` only removes the escape character. Macro parameter characters are doubled by `\tl_to_str:n`, and will also yield a non-empty result, hence are not considered as characters.

```

1745 \cs_new_protected:Npn \__cmd_grab_v_token_if_char:NTF #1
1746 { \str_if_eq:eeTF { } { \str_tail:n {#1} } }

```

(End definition for `__cmd_grab_v_token_if_char:NTF.`)

`__cmd_add_arg:n` When an argument is found it is stored, then further arguments are grabbed by calling `\l__cmd_signature_tl`.

```

1747 \cs_new_protected:Npn \__cmd_add_arg:n #1
1748 {
1749     \tl_put_right:Nn \l__cmd_args_tl { {#1} }
1750     \l__cmd_signature_tl \__cmd_run_code:
1751 }
1752 \cs_generate_variant:Nn \__cmd_add_arg:n { V , o , x }

```

(End definition for `__cmd_add_arg:n.`)

1.9 Grabbing arguments expandably

```
\_\_cmd\_expandable\_grab\_D:w
  \_\_cmd\_expandable\_grab\_D:NNNwNNnN
  \_\_cmd\_expandable\_grab\_D:NNNwNNnnN
\_\_cmd\_expandable\_grab\_D:Nw
  \_\_cmd\_expandable\_grab\_D:nnNNNwNN
```

The first step is to grab the first token or group. The generic grabbers $\langle function \rangle_{\sqcup}$ and $\langle function \rangle_{\sqcap}$ are just after $\backslash q_cmd$, we go and find them (and use the long one).

```
1753 \cs_new:Npn \_\_cmd\_expandable\_grab\_D:w #1 \q\_cmd #2#3
1754 { #2 { \_\_cmd\_expandable\_grab\_D:NNNwNNn #1 \q\_cmd #2 #3 } }
```

We then wish to test whether #7, which we just grabbed, is exactly #2. A preliminary test is whether their string representations coincide, then expand the only grabber function we have, #1, once: the two strings below are equal if and only if #7 matches #2 exactly.² The preliminary test is needed as #7 could validly contain $\backslash par$ (because a later mandatory argument could be long) and our grabber may be short. If #7 does not match #2, then the optional argument is missing, we use the default $-NoValue-$, and put back the argument #7 in the input stream.

If it does match, then interesting things need to be done. We will grab the argument piece by piece, with the following pattern:

```
<grabber> {<tokens>}
\q_nil {<piece 1>} <piece 2> \ERROR \q_cmd
\q_nil <input stream>
```

The $\langle grabber \rangle$ will find an opening delimiter in $\langle piece 2 \rangle$, take the $\backslash q_cmd$ as a second delimiter, and find more material delimited by the closing delimiter in the $\langle input stream \rangle$. We then move the part before the opening delimiter from $\langle piece 2 \rangle$ to $\langle piece 1 \rangle$, and the material taken from the $\langle input stream \rangle$ to the $\langle piece 2 \rangle$. Thus, the argument moves gradually from the $\langle input stream \rangle$ to the $\langle piece 2 \rangle$, then to the $\langle piece 1 \rangle$ when we have made sure to find all opening and closing delimiters. This two-step process ensures that nesting works: the number of opening delimiters minus closing delimiters in $\langle piece 1 \rangle$ is always equal to the number of closing delimiters in $\langle piece 2 \rangle$. We stop grabbing arguments once the $\langle piece 2 \rangle$ contains no opening delimiter any more, hence the balance is reached, and the final argument is $\langle piece 1 \rangle \langle piece 2 \rangle$. The indirection via $\backslash q_cmd_tmp:w$ allows to insert $-NoValue-$ expanded.

```
1755 \cs_set_protected:Npn \_\_cmd\_tmp:w #1
1756 {
1757   \cs_new:Npn \_\_cmd\_expandable\_grab\_D:NNNwNNn ##1##2##3##4 \q_cmd ##5##6##7
1758   {
1759     \str_if_eq:nnTF {##2} {##7}
1760     {
1761       \str_if_eq:onTF
1762         { ##1 { } { } ##7 ##2 \q_cmd ##3 }
1763         { { } {##2} { } }
1764     }
1765     { \use_i:nn }
1766   {
1767     ##1
1768     { \_\_cmd\_expandable\_grab\_D:NNNwNNnn ##1##2##3##4 \q_cmd ##5##6 }
```

²It is obvious that if #7 matches #2 then the strings are equal. We must check the converse. The right-hand-side of $\backslash str_if_eq:onTF$ does not end with #3, implying that the grabber function took everything as its arguments. The first brace group can only be empty if #7 starts with #2, otherwise the brace group preceding #7 would not vanish. The third brace group is empty, thus the $\backslash q_cmd$ that was used by our grabber #1 must be the one that we inserted (not some token in #7), hence the second brace group contains the end of #7 followed by #2. Since this is #2 on the right-hand-side, and no brace can be lost there, #7 must contain nothing else than its leading #2.

```

1769         \q_nil { } ##2 \ERROR \q__cmd \ERROR
1770     }
1771     { ##4 {#1} \q__cmd ##5 ##6 {##7} }
1772   }
1773 }
1774 \exp_args:No \__cmd_tmp:w { \c_novalue_t1 }

At this stage, #7 is  $\q_\text{nil} \{ \langle \text{piece 1} \rangle \} \langle \text{more for piece 1} \rangle$ , and we want to concatenate all that, removing  $\q_\text{nil}$ , and keeping the opening delimiter #2. Simply use  $\use_{\text{ii:nn}}$ . Also, #8 is  $\langle \text{remainder of piece 2} \rangle \text{\ERROR}$ , and #9 is  $\text{\ERROR} \langle \text{more for piece 2} \rangle$ . We concatenate those, replacing the two  $\text{\ERROR}$  by the closing delimiter #3.

```

```

1775 \cs_new:Npn \__cmd_expandable_grab_D:NNNwNNnnn #1#2#3#4 \q__cmd #5#6#7#8#9
1776 {
1777   \exp_args:Nof \__cmd_expandable_grab_D:nnNNNwNN
1778   { \use_{\text{ii:nn}} #7 #2 }
1779   { \__cmd_expandable_grab_D:Nw #3 \exp_stop_f: #8 #9 }
1780   #1#2#3 #4 \q__cmd #5 #6
1781 }
1782 \cs_new:Npn \__cmd_expandable_grab_D:Nw #1#2 \ERROR \ERROR { #2 #1 }


```

Armed with our two new $\langle \text{pieces} \rangle$, we are ready to loop. However, we must first see if $\langle \text{piece 2} \rangle$ (here #2) contains any opening delimiter #4. Again, we expand #3, this time removing its whole output with $\use_{\text{none:nnn}}$. The test is similar to \tl_if_in:nnTF . The token list is empty if and only if #2 does not contain the opening delimiter. In that case, we are done, and put the argument (from which we remove a spurious pair of delimiters coming from how we started the loop). Otherwise, we go back to looping with $__cmd_expandable_grab_D:NNNwNNnnn$. The code to deal with brace stripping is much the same as for the non-expandable case.

```

1783 \cs_new:Npn \__cmd_expandable_grab_D:nnNNNwNN #1#2#3#4#5#6 \q__cmd #7#8
1784 {
1785   \exp_args:No \tl_if_empty:oTF
1786   { #3 { \use_{\text{none:nnn}} } #2 \q__cmd #5 #4 \q__cmd #5 }
1787   {
1788     \tl_if_blank:oTF { \use_{\text{none:nn}} #1#2 }
1789     { \__cmd_put_arg_expandable:ow { \use_{\text{none:nn}} #1#2 } }
1790     {
1791       \str_if_eq:eeTF
1792       { \exp_not:o { \use_{\text{none:nn}} #1#2 } }
1793       { { \exp_not:o { \use_{\text{iii:nnnn}} #1#2 \q_\text{nil} } } }
1794       { \__cmd_put_arg_expandable:ow { \use_{\text{iii:nnnn}} #1#2 } }
1795       { \__cmd_put_arg_expandable:ow { \use_{\text{none:nn}} #1#2 } }
1796     }
1797     #6 \q__cmd #7 #8
1798   }
1799   {
1800     #3
1801     { \__cmd_expandable_grab_D:NNNwNNnnn #3#4#5#6 \q__cmd #7 #8 }
1802     \q_\text{nil} {#1} #2 \ERROR \q__cmd \ERROR
1803   }
1804 }


```

(End definition for $__cmd_expandable_grab_D:w$ and others.)

`_cmd_expandable_grab_D_alt:w`
`_cmd_expandable_grab_D_alt>NNwNNn`
`_cmd_expandable_grab_D_alt:Nwn`

When the delimiters are identical, nesting is not possible and a simplified approach is used. The test concept here is the same as for the case where the delimiters are different but there cannot be any nesting.

```

1805 \cs_new:Npn \_cmd_expandable_grab_D_alt:w #1 \q__cmd #2#3
1806   { #2 { \_cmd_expandable_grab_D_alt>NNwNNn #1 \q__cmd #2 #3 } }
1807 \cs_set_protected:Npn \_cmd_tmp:w #1
1808   {
1809     \cs_new:Npn \_cmd_expandable_grab_D_alt>NNwNNn ##1##2##3 \q__cmd ##4##5##6
1810     {
1811       \str_if_eq:nnTF {##6} {##2}
1812       {
1813         \str_if_eq:onTF
1814           { ##1 { } ##6 ##2 ##2 }
1815           { { } ##2 }
1816       }
1817       { \use_i:nn }
1818       {
1819         ##1
1820         { \_cmd_expandable_grab_D_alt>NNwn ##4 ##5 ##3 \q__cmd }
1821         ##6 \ERROR
1822       }
1823       { ##3 {#1} \q__cmd ##4 ##5 {##6} }
1824     }
1825   }
1826 \exp_args:No \_cmd_tmp:w { \c_novalue_tl }
1827 \cs_new:Npn \_cmd_expandable_grab_D_alt>NNwn #1#2#3 \q__cmd #4
1828   {
1829     \tl_if_blank:oTF { \use_none:n #4 }
1830     { \_cmd_put_arg_expandable:ow { \use_none:n #4 } }
1831     {
1832       \str_if_eq:eeTF
1833         { \exp_not:o { \use_none:n #4 } }
1834         { { \exp_not:o { \use_i:nn #4 \q_nil } } }
1835         { \_cmd_put_arg_expandable:ow { \use_i:nn #4 } }
1836         { \_cmd_put_arg_expandable:ow { \use_none:n #4 } }
1837     }
1838     #3 \q__cmd #1 #2
1839   }

```

(End definition for `_cmd_expandable_grab_D_alt:w`, `_cmd_expandable_grab_D_alt>NNwNNn`, and `_cmd_expandable_grab_D_alt:Nwn`)

```

\_\_cmd_expandable_grab_E:w
  \_\_cmd_expandable_grab_E_long:w
  \_\_cmd_expandable_grab_E_aux:w
  \_\_cmd_expandable_grab_E_test:nnw
\_\_cmd_expandable_grab_E_loop:nnnNNw
  \_\_cmd_expandable_grab_E_find:w
\_\_cmd_expandable_grab_E_find:nnw
  \_\_cmd_expandable_grab_E_end:nnw

```

We keep track of long/short by placing the appropriate grabber as the third token after `\q__cmd`; it is eventually removed by the `end:nnw` auxiliary. The `aux:w` auxiliary will be called repeatedly with two arguments: the set of pairs $\langle parser \rangle \langle token \rangle$, and the set of arguments found so far (initially all $\{-\text{NoValue}-\}$). At each step, grab what follows in the input stream then call the `loop:nnnNNw` auxiliary to compare it with each possible embellishment in turn. This auxiliary's #1 is what was found in the input, #2 collects $\langle parser \rangle \langle token \rangle$ pairs that did not match, #3 collects the corresponding arguments found previously, #4 and #5 is the current pair, #6 is the remaining pairs, #7 is empty or two `\q_nil`, and #8 is the current argument. If none of the pairs matched (determined by `\quark_if_nil:NTF`) then call the `end` auxiliary to stop looking for embellishments, remembering to put what was grabbed in the input back where it belongs, and storing

the arguments found just before `\q__cmd`. If the current argument #8 is not `-NoValue-` or if the input #1 does not match #5 (see t-type arguments below for a similar `\str_if_eq:onTF` test) then carry on the loop. Otherwise, we found a new embellishment: grab the corresponding argument in the input using the `find:w` auxiliary. To avoid losing braces around that auxiliary's argument we include a space, which will be eliminated in the next loop through embellishments.

```

1840 \cs_new:Npn \__cmd_expandable_grab_E:w #1 \q__cmd #2#3
1841   { \__cmd_expandable_grab_E_aux:w #1 \q__cmd #2 #3 #3 }
1842 \cs_new:Npn \__cmd_expandable_grab_E_long:w #1 \q__cmd #2#3
1843   { \__cmd_expandable_grab_E_aux:w #1 \q__cmd #2 #3 #2 }
1844 \cs_new:Npn \__cmd_expandable_grab_E_aux:w #1 \q__cmd #2#3#4
1845   { #2 { \__cmd_expandable_grab_E_test:nw #1 \q__cmd #2 #3 #4 } }
1846 \cs_new:Npn \__cmd_expandable_grab_E_test:nw #1#2#3 \q__cmd #4#5#6#7
1847   {
1848     \__cmd_expandable_grab_E_loop:nnnNNw {#7} { } { }
1849       #1 \q_nil \q_nil \q_mark #2 \q_nil
1850       #3 \q__cmd #4 #5 #6
1851   }
1852 \cs_new:Npn \__cmd_expandable_grab_E_loop:nnnNNw
1853   #1#2#3#4#5#6 \q_nil #7 \q_mark #8
1854   {
1855     \quark_if_nil:NTF #4
1856       { \__cmd_expandable_grab_E_end:nw {#1} {#3} }
1857       {
1858         \tl_if_novalue:nTF {#8}
1859           { \str_if_eq:onTF { #4 { } #1 #5 } {#5} }
1860           { \use_ii:nn }
1861             { \__cmd_expandable_grab_E_find:w { #2 #4 #5 #6 } {#3} ~ }
1862             {
1863               \__cmd_expandable_grab_E_loop:nnnNNw
1864                 {#1} { #2 #4 #5 } { #3 {#8} }
1865                 #6 \q_nil #7 \q_mark
1866             }
1867       }
1868   }
1869 \cs_new:Npn \__cmd_expandable_grab_E_find:w #1 \q__cmd #2#3#4
1870   { #4 { \__cmd_expandable_grab_E_find:nw #1 \q__cmd #2 #3 #4 } }
1871 \cs_new:Npn \__cmd_expandable_grab_E_find:nw #1#2#3 \q_nil #4 \q__cmd #5#6#7#8
1872   { \__cmd_expandable_grab_E_aux:w {#1} { #2 {#8} #3 } #4 \q__cmd #5 #6 #7 }
1873 \cs_new:Npn \__cmd_expandable_grab_E_end:nw #1#2#3 \q__cmd #4#5#6
1874   { #3 #2 \q__cmd #4 #5 {#1} }

(End definition for \__cmd_expandable_grab_E:w and others.)
```

The mandatory case is easy: find the auxiliary after the `\q__cmd`, and use it directly to grab the argument, then correctly position the argument before `\q__cmd`.

```

1875 \cs_new:Npn \__cmd_expandable_grab_m:w #1 \q__cmd #2#3
1876   { #3 { \__cmd_expandable_grab_m_aux:wNn #1 \q__cmd #2 #3 } }
1877 \cs_new:Npn \__cmd_expandable_grab_m_long:w #1 \q__cmd #2#3
1878   { #2 { \__cmd_expandable_grab_m_aux:wNn #1 \q__cmd #2 #3 } }
1879 \cs_new:Npn \__cmd_expandable_grab_m_aux:wNn #1 \q__cmd #2#3#4
1880   { #1 {#4} \q__cmd #2 #3 }
```

(End definition for `__cmd_expandable_grab_m:w`, `__cmd_expandable_grab_m_long:w`, and `__cmd_expandable_grab_m_aux:wNn`.)

```

\__cmd_expandable_grab_R:w Much the same as for the D-type argument, with only the lead-off function varying.
\__cmd_expandable_grab_R_aux:NNNwNNn
1881 \cs_new:Npn \__cmd_expandable_grab_R:w #1 \q__cmd #2#3
1882 { #2 { \__cmd_expandable_grab_R_aux:NNNwNNn #1 \q__cmd #2#3 } }
1883 \cs_set_protected:Npn \__cmd_tmp:w #1
1884 {
1885     \cs_new:Npn \__cmd_expandable_grab_R_aux:NNNwNNn ##1##2##3##4 \q__cmd ##5##6##7
1886     {
1887         \str_if_eq:nnTF {##7} {##2}
1888         {
1889             \str_if_eq:onTF
1890             { ##1 { } { } ##7 ##2 \q__cmd ##3 }
1891             { { } {##2} { } }
1892         }
1893         { \use_i:nn }
1894         {
1895             ##1
1896             { \__cmd_expandable_grab_D:NNNwNNnnn ##1##2##3##4 \q__cmd ##5##6 }
1897             \q_nil { } ##2 \ERROR \q__cmd \ERROR
1898         }
1899         {
1900             \msg_expandable_error:nnff { cmd } { missing-required }
1901             { \exp_args:Nf \tl_trim_spaces:n { \token_to_str:N ##5 } }
1902             { \tl_to_str:n {##2} }
1903             ##4 {#1} \q__cmd ##5 ##6 {##7}
1904         }
1905     }
1906 }
1907 \exp_args:No \__cmd_tmp:w { \c_novalue_tl }

(End definition for \__cmd_expandable_grab_R:w and \__cmd_expandable_grab_R_aux:NNNwNNn.)

```

```

\__cmd_expandable_grab_R_alt:w When the delimiters are identical, nesting is not possible and a simplified approach is
\__cmd_expandable_grab_R_alt_aux:NNNwNNn used. The test concept here is the same as for the case where the delimiters are different.
1908 \cs_new:Npn \__cmd_expandable_grab_R_alt:w #1 \q__cmd #2#3
1909 { #2 { \__cmd_expandable_grab_R_alt_aux:NNNwNNn #1 \q__cmd #2#3 } }
1910 \cs_set_protected:Npn \__cmd_tmp:w #1
1911 {
1912     \cs_new:Npn \__cmd_expandable_grab_R_alt_aux:NNNwNNn ##1##2##3 \q__cmd ##4##5##6
1913     {
1914         \str_if_eq:nnTF {##6} {##2}
1915         {
1916             \str_if_eq:onTF
1917             { ##1 { } ##6 ##2 ##2 }
1918             { { } ##2 }
1919         }
1920         { \use_i:nn }
1921         {
1922             ##1
1923             { \__cmd_expandable_grab_D_alt:NNNwNNn ##4 ##5 ##3 \q__cmd }
1924             ##6 \ERROR
1925         }
1926     }
1927     \msg_expandable_error:nnff { cmd } { missing-required }
1928     { \exp_args:Nf \tl_trim_spaces:n { \token_to_str:N ##4 } }

```

```

1929           { \tl_to_str:n {##2} }
1930           ##3 {#1} \q_cmd ##4 ##5 {##6}
1931       }
1932   }
1933 }
1934 \exp_args:No \__cmd_tmp:w { \c_novalue_tl }

(End definition for \__cmd_expandable_grab_R_alt:w and
\__cmd_expandable_grab_R_alt_aux:NNwNNn.)

```

As for a D-type argument, here we compare the grabbed tokens using the only parser we have in order to work out if #2 is exactly equal to the output of the grabber.

```

1935 \cs_new:Npn \__cmd_expandable_grab_t:w #1 \q_cmd #2#3
1936   { #2 { \__cmd_expandable_grab_t_aux:NNwm #1 \q_cmd #2 #3 } }
1937 \cs_new:Npn \__cmd_expandable_grab_t_aux:NNwn #1#2#3 \q_cmd #4#5#6
1938   {
1939     \str_if_eq:onTF { #1 { } #6 #2 } {#2}
1940     { #3 { \BooleanTrue } \q_cmd #4 #5 }
1941     { #3 { \BooleanFalse } \q_cmd #4 #5 {#6} }
1942   }

```

(End definition for __cmd_expandable_grab_t:w and __cmd_expandable_grab_t_aux:NNwn.)

A useful helper, to store arguments when they are ready.

```

1943 \cs_new:Npn \__cmd_put_arg_expandable:nw #1#2 \q_cmd { #2 {#1} \q_cmd }
1944 \cs_generate_variant:Nn \__cmd_put_arg_expandable:nw { o }

```

(End definition for __cmd_put_arg_expandable:nw.)

1.10 Argument processors

__cmd_bool_reverse:N A simple reversal.

```

1945 \cs_new_protected:Npn \__cmd_bool_reverse:N #1
1946   {
1947     \bool_if:NTF #1
1948     { \tl_set:Nn \ProcessedArgument { \c_false_bool } }
1949     { \tl_set:Nn \ProcessedArgument { \c_true_bool } }
1950   }

```

(End definition for __cmd_bool_reverse:N.)

```
\l__cmd_split_list_seq
```

```
\l__cmd_split_list_tl
```

Splitting can take place either at a single token or at a longer identifier. To deal with single active tokens, a two-part procedure is needed.

```
\l__cmd_split_list_multi:nV
\l__cmd_split_list_single:Nn
1951 \seq_new:N \l__cmd_split_list_seq
1952 \tl_new:N \l__cmd_split_list_tl
1953 \cs_new_protected:Npn \l__cmd_split_list:nn #1#2
1954 {
1955     \tl_if_single:nTF {#1}
1956     {
1957         \token_if_cs:NTF #1
1958         { \l__cmd_split_list_multi:nn {#1} {#2} }
1959         { \l__cmd_split_list_single:Nn #1 {#2} }
1960     }
1961     { \l__cmd_split_list_multi:nn {#1} {#2} }
1962 }
1963 \cs_new_protected:Npn \l__cmd_split_list_multi:nn #1#2
1964 {
1965     \seq_set_split:Nnn \l__cmd_split_list_seq {#1} {#2}
1966     \tl_clear:N \ProcessedArgument
1967     \seq_map_inline:Nn \l__cmd_split_list_seq
1968     { \tl_put_right:Nn \ProcessedArgument { {##1} } }
1969 }
1970 \cs_generate_variant:Nn \l__cmd_split_list_multi:nn { nV }
1971 \group_begin:
1972 \char_set_catcode_active:N \^^@
```

```
\cs_new_protected:Npn \l__cmd_split_list_single:Nn #1#2
```

```
{
1974     \tl_set:Nn \l__cmd_split_list_tl {#2}
1975     \group_begin:
1976     \char_set_lccode:nn { '\^^@ } { '#1 }
1977     \tex_lowercase:D
1978     {
1979         \group_end:
1980         \group_end:
1981         \tl_replace_all:Nnn \l__cmd_split_list_tl { ^@ }
1982         {#1}
1983         \l__cmd_split_list_multi:nV {#1} \l__cmd_split_list_tl
1984     }
1985 }
```

```
\group_end:
```

(End definition for `\l__cmd_split_list:nn`, `\l__cmd_split_list_multi:nn`, and
`\l__cmd_split_list_single:Nn`.)

```
\l__cmd_split_argument:nnn
```

```
\l__cmd_split_argument_aux:nnnn
```

Splitting to a known number of items is a special version of splitting a list, in which the limit is hard-coded and where there will always be exactly the correct number of output items. An auxiliary function is used to save on working out the token list length several times.

```
1986 \cs_new_protected:Npn \l__cmd_split_argument:nnn #1#2#3
1987 {
1988     \l__cmd_split_list:nn {#2} {#3}
1989     \exp_args:Nf \l__cmd_split_argument_aux:nnnn
1990     { \tl_count:N \ProcessedArgument }
1991     {#1} {#2} {#3}
1992 }
```

```

1993 \cs_new_protected:Npn \__cmd_split_argument_aux:nnnn #1#2#3#4
1994 {
1995   \int_compare:nNnF {#1} = { #2 + 1 }
1996   {
1997     \int_compare:nNnTF {#1} > { #2 + 1 }
1998     {
1999       \tl_set:Nx \ProcessedArgument
2000       {
2001         \exp_last_unbraced:NnNo
2002           \__cmd_split_argument_aux:n
2003           { #2 + 1 }
2004         \use_none_delimit_by_q_stop:w
2005         \ProcessedArgument
2006         \q_stop
2007       }
2008     \msg_error:nxxxx { cmd } { arg-split }
2009     { \tl_to_str:n {#3} } { \int_eval:n { #2 + 1 } }
2010     { \tl_to_str:n {#4} }
2011   }
2012   {
2013     \tl_put_right:Nx \ProcessedArgument
2014     {
2015       \prg_replicate:nn { #2 + 1 - (#1) }
2016       { { \exp_not:V \c_novalue_tl } }
2017     }
2018   }
2019 }
2020 }
```

Auxiliaries to leave exactly the correct number of arguments in \ProcessedArgument.

```

2021 \cs_new:Npn \__cmd_split_argument_aux:n #1
2022   { \prg_replicate:nn {#1} { \__cmd_split_argument_aux:wn } }
2023 \cs_new:Npn \__cmd_split_argument_aux:wn #1 \use_none_delimit_by_q_stop:w #2
2024   {
2025     \exp_not:n { {#2} }
2026     #1
2027     \use_none_delimit_by_q_stop:w
2028   }
```

(End definition for __cmd_split_argument:nnn and others.)

__cmd_trim_spaces:n This one is almost trivial.

```

2029 \cs_new_protected:Npn \__cmd_trim_spaces:n #1
2030   { \tl_set:Nx \ProcessedArgument { \tl_trim_spaces:n {#1} } }
```

(End definition for __cmd_trim_spaces:n.)

1.11 Access to the argument specification

__cmd_get_arg_spec_error:N
__cmd_get_arg_spec_error:n
__cmd_get_arg_spec_error_aux:N

```

2031 \cs_new_protected:Npn \__cmd_get_arg_spec_error:N #1
2032   {
2033     \bool_set_false:N \l__cmd_environment_bool
```

```

2034     \tl_set:Nn \l__cmd_fn_tl {#1}
2035     \__cmd_get_arg_spec_error_aux:n { \cs_if_exist:NTF #1 }
2036   }
2037 \cs_new_protected:Npn \__cmd_get_arg_spec_error:n #1
2038   {
2039     \bool_set_true:N \l__cmd_environment_bool
2040     \str_set:Nx \l__cmd_environment_str {#1}
2041     \__cmd_get_arg_spec_error_aux:n
2042       { \cs_if_exist:cTF { \l__cmd_environment_str } }
2043   }
2044 \cs_new_protected:Npn \__cmd_get_arg_spec_error_aux:n #1
2045   {
2046     #1
2047   {
2048     \msg_error:nnx { cmd } { non-xparse }
2049       { \__cmd_environment_or_command: }
2050   }
2051   {
2052     \msg_error:nnx { cmd } { unknown }
2053       { \__cmd_environment_or_command: }
2054   }
2055 }
```

(End definition for __cmd_get_arg_spec_error:N, __cmd_get_arg_spec_error:n, and __cmd_get_arg_spec_error_aux:n.)

__cmd_get_arg_spec:NTF If the command is not an `xparse` command, complain. If it is, its second “item” is the argument specification.

```

2056 \cs_new_protected:Npn \__cmd_get_arg_spec:NTF #1#2#3
2057   {
2058     \__kernel_cmd_if_xparse:NTF #1
2059   {
2060     \tl_set:Nx \ArgumentSpecification { \tl_item:Nn #1 { 2 } }
2061     #2
2062   }
2063   {#3}
2064 }
```

(End definition for __cmd_get_arg_spec:NTF.)

Rolling forward from 2020-10-01 is tricky because the entire `ltcmd` module is new, but the user-level commands have the same name, so only these will clash. To work around that, in `latexrelease` mode we will (temporarily) disable `__kernel_chk_if_free_cs:N` for this final part of the code, then restore at the end.

```

2065 <|latexrelease>\cs_new_eq:NN \__cmd_chk_if_free_cs:N \__kernel_chk_if_free_cs:N
2066 <|latexrelease>\cs_gset_eq:NN \__kernel_chk_if_free_cs:N \use_none:n
```

\ArgumentSpecification 2067 \tl_new:N \ArgumentSpecification

__cmd_get_arg_spec:N Recovering the argument specification is now trivial.

```

2068 \cs_new_protected:Npn \__cmd_get_arg_spec:N #1
2069   {
2070     \__cmd_get_arg_spec:NTF #1 { }
```

```

2071      { \__cmd_get_arg_spec_error:N #1 }
2072  }
2073 \cs_new_protected:Npn \__cmd_get_arg_spec:n #1
2074  {
2075      \exp_args:Nc \__cmd_get_arg_spec:NTF
2076      { environment~ \tl_to_str:n {#1} }
2077      { }
2078      { \__cmd_get_arg_spec_error:n {#1} }
2079  }

```

(End definition for `__cmd_get_arg_spec:N` and `__cmd_get_arg_spec:n`.)

`__cmd_show_arg_spec:N` Showing the argument specification simply means finding it and then calling the `\tl_show:N` function.

```

2080 \cs_new_protected:Npn \__cmd_show_arg_spec:N #1
2081  {
2082      \__cmd_get_arg_spec:NTF #1
2083      { \tl_show:N \ArgumentSpecification }
2084      { \__cmd_get_arg_spec_error:N #1 }
2085  }
2086 \cs_new_protected:Npn \__cmd_show_arg_spec:n #1
2087  {
2088      \exp_args:Nc \__cmd_get_arg_spec:NTF
2089      { environment~ \tl_to_str:n {#1} }
2090      { \tl_show:N \ArgumentSpecification }
2091      { \__cmd_get_arg_spec_error:n {#1} }
2092  }

```

(End definition for `__cmd_show_arg_spec:N` and `__cmd_show_arg_spec:n`.)

1.12 Utilities

`__cmd_check_definable:nNT` Check that a token list is appropriate as a first argument of `\NewDocumentCommand` and similar functions and otherwise produce an error. First trim whitespace to allow for spaces around the actual command to be defined. If the result has multiple tokens, it is not a valid argument. The single token is a control sequence exactly if its string representation has more than one character (using `\token_to_str:N` rather than `\tl_to_str:n` to avoid problems with macro parameter characters, and setting `\tex_escapechar:D` to prevent it from being non-printable). Finally, check for an active character: this is done by lowercasing the token to fix its character code (arbitrarily to that of `?`) and comparing the result to an active `?`. Both control sequences and active characters are valid arguments, and non-active character tokens are not. In all cases, the group opened to keep assignments local must be closed.

```

2093 \cs_new_protected:Npn \__cmd_check_definable:nNT #1
2094  { \tl_trim_spaces_apply:nN {#1} \__cmd_check_definable_aux:nN }
2095 \group_begin:
2096  \char_set_catcode_active:n { ‘? ’ }
2097 \cs_new_protected:Npn \__cmd_check_definable_aux:nN #1#2
2098  {
2099  \group_begin:
2100  \tl_if_single_token:nTF {#1}
2101  {
2102      \int_set:Nn \tex_escapechar:D { 92 }

```

```

2103 \exp_args:Nx \tl_if_empty:nTF
2104     { \exp_args:No \str_tail:n { \token_to_str:N #1 } }
2105     {
2106         \exp_args:Nx \char_set_lccode:nn
2107             { ` \str_head:n {#1} } { ? }
2108         \tex_lowercase:D { \tl_if_eq:nnTF {#1} } { ? }
2109             { \group_end: \use_iii:nnn }
2110             { \group_end: \use_i:nnn }
2111     }
2112     { \group_end: \use_iii:nnn }
2113 }
2114 { \group_end: \use_ii:nnn }
2115 {
2116     \msg_error:nnxx { cmd } { not-definable }
2117         { \tl_to_str:n {#1} } { \token_to_str:N #2 }
2118 }
2119 {
2120     \msg_error:nnxx { cmd } { not-one-token }
2121         { \tl_to_str:n {#1} } { \token_to_str:N #2 }
2122 }
2123 }
2124 \group_end:

```

(End definition for `_cmd_check_definable:nNT` and `_cmd_check_definable_aux:nN.`)

`_cmd_token_if_cs:NTF` Based on the definition of `_cmd_check_definable_aux:nN` above, but only checks for an actual control sequence (*i.e.*, `\langle anything \rangle`). `\tex_escapechar:D` is temporarily changed to a known value and then it checks if `\string#1` contains more than one character: if it does, it's a control sequence. This test differs from `\token_if_cs:NTF` for example in `\token_if_cs:NTF \c_group_begin_token {T}{F}`, where `\token_if_-cs:NTF` returns false.

```
2125 \cs_new_protected:Npn \__cmd_token_if_cs:NTF #1
2126 {
2127     \group_begin:
2128     \int_set:Nn \tex_escapechar:D { 92 }
2129     \exp_args:Nx \tl_if_empty:nTF
2130         { \exp_args:No \str_tail:n { \token_to_str:N #1 } }
2131         { \group_end: \use_ii:nn }
2132         { \group_end: \use_i:nn }
2133 }
```

(End definition for `_cmd_token_if_cs:NTF`.)

Analogue of \seq_mapthread_function:NNN for token lists.

```
\_cmd_t1_mapthread_function:nnN 2134 \cs_new:Npn \__cmd_t1_mapthread_function:NNN #1#2#3
\__cmd_t1_mapthread_loop:w 2135 {
 2136   \exp_after:wN \exp_after:wN
 2137   \exp_after:wN \__cmd_t1_mapthread_loop:w
 2138   \exp_after:wN \exp_after:wN
 2139   \exp_after:wN #3
 2140   \exp_after:wN #1
 2141   \exp_after:wN \q_recursion_tail
 2142   \exp_after:wN \q_mark
 2143   #2
```

```

2144      \q_recursion_tail
2145      \q_recursion_stop
2146    }
2147 \cs_new:Npn \__cmd_tl_mapthread_function:nnN #1#2#3
2148  {
2149    \__cmd_tl_mapthread_loop:w #3
2150    #1 \q_recursion_tail \q_mark
2151    #2 \q_recursion_tail \q_recursion_stop
2152  }
2153 \cs_new:Npn \__cmd_tl_mapthread_loop:w #1#2#3 \q_mark #4
2154  {
2155    \quark_if_recursion_tail_stop:n {#2}
2156    \quark_if_recursion_tail_stop:n {#4}
2157    #1 {#2} {#4}
2158    \__cmd_tl_mapthread_loop:w #1#3 \q_mark
2159  }

(End definition for \__cmd_tl_mapthread_function:NNN, \__cmd_tl_mapthread_function:nnN, and
\__cmd_tl_mapthread_loop:w.)
```

__kernel_cmd_if_xparse:NTF
__cmd_cmd_type_cases:Nnnnn
__cmd_cmd_if_xparse_aux:N

To determine whether the command is an `xparse` command check that its `arg_spec` is empty (this also excludes non-macros) and that its `replacement_spec` starts with either `__cmd_start:nNNnnn` (non-expandable command) or `__cmd_start_expandable:nNNNNn` (expandable command) or `__cmd_start_env:nnnnn` (environment) or `\environment #1 end aux` (environment end).

This conditional is needed in several kernel modules and is therefore has a kernel-internal name.

```

2160 \cs_new_protected:Npn \__cmd_cmd_type_cases:NnnnnF #1 #2 #3 #4 #5 #6
2161  {
2162    \exp_args:Ne \str_case_e:nnF
2163    {
2164      \exp_args:Nf \tl_if_empty:nT { \cs_argument_spec:N #1 }
2165      {
2166        \token_if_macro:NT #1
2167        {
2168          \exp_after:wN \exp_after:wN
2169          \exp_after:wN \token_to_str:N
2170          \exp_after:wN \use_i_delimit_by_q_stop:nw
2171          #1 \scan_stop: \q_stop
2172        }
2173      }
2174    {
2175      {
2176        { \token_to_str:N \__cmd_start:nNNnnn } {#2}
2177        { \token_to_str:N \__cmd_start_expandable:nNNNNn } {#3}
2178        { \token_to_str:N \__cmd_start_env:nnnnn } {#4}
2179        {
2180          \exp_after:wN \token_to_str:N
2181          \cs:w environment-
2182          \exp_last_unbraced:Ne \use_none:nnn
2183          { \cs_to_str:N #1 } ~end-aux \cs_end:
2184        } {#5}
2185      }
2186    {#6}
```

```

2187     }
2188 \cs_new_protected:Npn \_kernel_cmd_if_xparse:NTF #1
2189   {
2190     \_cmd_cmd_type_cases:NnnnnF #1
2191     { } { } { } { } { \use_iii:n }
2192     \use_i:nn
2193   }

(End definition for \_kernel_cmd_if_xparse:NTF, \_cmd_cmd_type_cases:Nnnnn, and
\_cmd_cmd_if_xparse_aux:N.)

```

_cmd_peek_nonspace:NTF Collect spaces in a loop, and put the collected spaces back in the false branch of a call to _peek_meaning:NTF or _peek_meaning_remove:NTF.

```

2194 \cs_new_protected:Npn \_cmd_peek_nonspace:NTF
2195   { \_cmd_peek_nonspace_aux:nNNTF { } \_cmd_peek_meaning:NTF }
2196 \cs_new_protected:Npn \_cmd_peek_nonspace_remove:NTF
2197   { \_cmd_peek_nonspace_aux:nNNTF { } \_cmd_peek_meaning_remove:NTF }
2198 \cs_new_protected:Npn \_cmd_peek_nonspace_aux:nNNTF #1#2#3#4#5
2199   {
2200     \_peek_meaning_remove:NTF \c_space_token
2201     { \_cmd_peek_nonspace_aux:nNNTF { #1 ~ } #2 #3 {#4} {#5} }
2202     { #2 #3 { #4 } { #5 #1 } }
2203   }

(End definition for \_cmd_peek_nonspace:NTF, \_cmd_peek_nonspace_remove:NTF, and
\_cmd_peek_nonspace_aux:nNNTF.)

```

_cmd_peek_meaning:NTF Peek ahead for a token with a given meaning. In case the search token is a control sequence, also check that the $\langle csname \rangle$ is the same as the control sequence peeked at. This extra verification is necessary when the command is delimited by control sequence tokens (as opposed to character tokens), and we want the exact same control sequence to match.

```

2204 \cs_new_protected:Npn \_cmd_peek_meaning:NTF
2205   { \_cmd_peek_meaning_aux:NNTF \c_false_bool }
2206 \cs_new_protected:Npn \_cmd_peek_meaning_remove:NTF
2207   { \_cmd_peek_meaning_aux:NNTF \c_true_bool }
2208 \cs_new_protected:Npn \_cmd_peek_meaning_aux:NNTF #1#2#3#4
2209   {
2210     \tl_set:Nn \l__cmd_tmpa_tl {#3}
2211     \tl_set:Nn \l__cmd_tmpb_tl {#4}
2212     \_peek_meaning:NTF #2
2213     {
2214       \token_if_eq_meaning:NNTF #2 \c_group_begin_token
2215       { \_cmd_peek_true_remove:Nw #1 }
2216     {
2217       \_cmd_token_if_cs:NTF #2
2218       { \_cmd_peek_cs_check_equal:NNN #1 #2 }
2219       { \_cmd_peek_true_remove:Nw #1 }
2220     }
2221   }
2222   { \l__cmd_tmpb_tl }
2223 }

\cs_new_protected:Npn \_cmd_peek_cs_check_equal:NNN #1#2#3
2224   {

```

```

2226     \str_if_eq:nnTF {#2} {#3}
2227     {
2228         \__cmd_peek_true_remove:Nw #1
2229         \l__cmd_tmpb_tl
2230     }
2231 \cs_new_protected:Npn \__cmd_peek_true_remove:Nw #1
2232 {
2233     \bool_if:NTF #1
2234     {
2235         \tex_afterassignment:D \l__cmd_tmpa_tl
2236         \cs_set_eq:NN \__cmd_tmp:w
2237     }
2238     \l__cmd_tmpa_tl
2239 }

```

(End definition for `__cmd_peek_meaning:NTF` and others.)

1.13 Messages

```

\c__cmd_ignore_def_tl 2240 \tl_const:Nn \c__cmd_ignore_def_tl
2241   { \\ \\ LaTeX-will-ignore-this-entire-definition. }

```

`__cmd_environment_or_command`: Two texts used in several messages.

```

2242 \cs_new:Npn \__cmd_environment_or_command:
2243 {
2244     \bool_if:NTF \l__cmd_environment_bool
2245     { environment ~ , \l__cmd_environment_str ~ }
2246     {
2247         command ~ ,
2248         \exp_args:Nf \tl_trim_spaces:n
2249         { \exp_after:wN \token_to_str:N \l__cmd_fn_tl }
2250     ,
2251 }
2252 }

```

(End definition for `__cmd_environment_or_command`.)

Some messages intended as errors when defining commands/environments.

```

2253 \msg_new:nnnn { cmd } { arg-after-body }
2254   { Argument-type-'b'-must-be-last-in-#1. }
2255   {
2256     The-'b'-argument-type-must-come-last-but-it-is-followed-
2257     by-'#2'-in-the-argument-specification.-This-is-not-allowed.
2258     \c__cmd_ignore_def_tl
2259   }
2260 \msg_new:nnnn { cmd } { bad-arg-spec }
2261   { Bad-argument-specification-'#2'-for-#1. }
2262   {
2263     The-argument-specification-provided-is-not-valid:-
2264     one-or-more-mandatory-parts-are-missing.
2265     \c__cmd_ignore_def_tl
2266   }

```

```

2267 \msg_new:nnnn { cmd } { already-defined }
2268   { Command-'#1'~already~defined. }
2269   {
2270     You~have~used~#2~
2271     with~a~command~that~already~has~a~definition. \\ \\
2272     The~existing~definition~of~'#1'~will~not~be~altered.
2273   }
2274 \msg_new:nnnn { cmd } { undefined }
2275   { Command ~'#1'~undefined. }
2276   {
2277     You~have~used~#2~
2278     with~a~command~that~was~never~defined.
2279     \c__cmd_ignore_def_tl
2280   }
2281 \msg_new:nnnn { cmd } { env-already-defined }
2282   { Environment~'#1'~already~defined. }
2283   {
2284     You~have~used~\NewDocumentEnvironment
2285     with~an~environment~that~already~has~a~definition. \\ \\
2286     The~existing~definition~of~'#1'~will~not~be~altered.
2287   }
2288 \msg_new:nnnn { cmd } { env-end-already-defined }
2289   { End~of~environment~'#1'~already~defined. }
2290   {
2291     You~have~used~\NewDocumentEnvironment
2292     with~an~environment~that~already~has~a~definition~for~'end#1'. \\ \\
2293     The~existing~definition~of~'#1'~will~not~be~altered.
2294   }
2295 \msg_new:nnnn { cmd } { env-undefined }
2296   { Environment~'#1'~undefined. }
2297   {
2298     You~have~used~\RenewDocumentEnvironment
2299     with~an~environment~that~was~never~defined.
2300     \c__cmd_ignore_def_tl
2301   }
2302 \msg_new:nnnn { cmd } { expandable-ending-optional }
2303   { Bad~argument~specification~'#2'~for~#1. }
2304   {
2305     Expandable~commands~must~have~a~final~mandatory~argument~
2306     (or~no~arguments~at~all).~You~cannot~have~a~terminal~optional~
2307     argument~with~expandable~commands.
2308   }
2309 \msg_new:nnnn { cmd } { long-short-mix }
2310   { Invalid~argument~prefix~'+'~in~command~'#1'. }
2311   {
2312     The~arguments~for~an~expandable~command~must~not~involve~short~
2313     arguments~after~long~arguments.~You~have~tried~to~mix~the~two~types~
2314     when~defining~'#1'.
2315   }
2316 \msg_new:nnnn { cmd } { invalid-command-arg }
2317   { Invalid~argument~type~'#2'~in~#1. }
2318   {
2319     The~letter~'#2'~can~only~be~used~in~environment~argument~
2320     specifications,~but~not~for~commands.

```

```

2321      \\ \\
2322      LaTeX-will-ignore-the-entire-definition.
2323  }
2324 \msg_new:nnnn { cmd } { invalid-expandable-arg }
2325   { Invalid-argument-type-'#2'-in-#1. }
2326   {
2327     The-letter-'#2'-specifies-an-argument-type-which-cannot-be-used-
2328     in-an-expandable-command.
2329     \c__cmd_ignore_def_tl
2330   }
2331 \msg_new:nnnn { cmd } { invalid-after-optional-expandably }
2332   { Argument-'#2'-invalid-after-optional-arg-in-#1. }
2333   {
2334     The-letter-'#2'-specifies-an-argument-type-which-cannot-be-used-
2335     in-an-expandable-command-after-an-optional-argument.
2336     \c__cmd_ignore_def_tl
2337   }
2338 \msg_new:nnnn { cmd } { invalid-bang }
2339   { Invalid-argument-prefix-'!'~in-#1. }
2340   {
2341     The-prefix-'!'~is-only-allowed-for-trailing-optional-arguments.~
2342     You-tried-to-apply-it-to-'#2'.
2343     \c__cmd_ignore_def_tl
2344   }
2345 \msg_new:nnnn { cmd } { not-definable }
2346   { First-argument-of-'#2'-must-be-a-command. }
2347   {
2348     The-first-argument-of-'#2'-should-be-the-document-command-that-will-
2349     be-defined.~The-provided-argument-'#1'-is-a-character.~Perhaps-a-
2350     backslash-is-missing?
2351     \c__cmd_ignore_def_tl
2352   }
2353 \msg_new:nnnn { cmd } { not-one-token }
2354   { First-argument-of-'#2'-must-be-a-command. }
2355   {
2356     The-first-argument-of-'#2'-should-be-the-document-command-that-will-
2357     be-defined.~The-provided-argument-'#1'-contains-more-than-one-
2358     token.~Perhaps-a-backslash-is-missing?
2359     \c__cmd_ignore_def_tl
2360   }
2361 \msg_new:nnnn { cmd } { not-single-token }
2362   { Argument-delimiter-'#2'-invalid-in-#1. }
2363   {
2364     The-argument-specification-contains-
2365     \tl_if_empty:nTF{#2}{nothing}{'#2'}-
2366     in-a-place-
2367     where-a-single-token-is-required.
2368     \c__cmd_ignore_def_tl
2369   }
2370 \msg_new:nnnn { cmd } { forbidden-group-token }
2371   { Argument-delimiter-'#2'-invalid-in-#1. }
2372   {
2373     The-argument-specification-contains-the-implicit-
2374     #3-group-token-'#2'-which-is-not-allowed-as-an-argument-delimiter.

```

```

2375      \c__cmd_ignore_def_tl
2376  }
2377 \msg_new:nnnn { cmd } { processor-in-expandable }
2378 { Invalid-argument-prefix->'~in-command-'#1'. }
2379 {
2380   The~argument~specification-for-'#1'~contains~the~processor~function-'>{#2}'.
2381   This~is~only~supported~for~robust~commands,~but~not~for~expandable~ones.
2382   \c__cmd_ignore_def_tl
2383 }
2384 \msg_new:nnnn { cmd } { too-many-args }
2385 { Too-many~arguments~for~#1. }
2386 {
2387   The~argument~specification-'#2'~asks~for~more~than~9~arguments.~
2388   This~cannot~be~implemented.
2389   \c__cmd_ignore_def_tl
2390 }
2391 \msg_new:nnnn { cmd } { two-markers }
2392 { Invalid-argument-prefix-'#2'~in~#1. }
2393 {
2394   The~argument~specification~provided~for~#1~has~two~'#2'~markers~applied~to~the~same~argument;~one~is~redundant.
2395 }
2396
2397 \msg_new:nnnn { cmd } { unknown-argument-type } % should be unkown-arg-type but dep in xparsen
2398 { Invalid-argument-type-'#2'~in~#1. }
2399 {
2400   The~letter~'#2'~does~not~specify~a~known~argument~type.
2401   \c__cmd_ignore_def_tl
2402 }
2403 \msg_new:nnnn { cmd } { xparsen-arg-type }
2404 { Invalid-argument-type-'#2'~in~#1~(requires~xparsen). }
2405 {
2406   The~letter~'#2'~specifies~a~known~but~deprecated~argument~type.~
2407   If~you~really~need~it~you~have~to~load~the~xparsen~package.
2408   \c__cmd_ignore_def_tl
2409 }

```

Errors when using commands/environments. The `if-boolean` message is always used in expandable errors. The `default-loop` and `missing-required` messages can be expandable or not expandable.

```

2410 \msg_new:nnn { cmd } { if-boolean }
2411 { Invalid-argument-{#1}~to~\iow_char:N\\IfBoolean... }
2412 \msg_new:nnnn { cmd } { default-loop }
2413 { Circular-dependency~in~defaults~of~#1. }
2414 {
2415   The~default~values~of~two~or~more~arguments~of~the~#1~depend~on~each~other~in~a~way~that~cannot~be~resolved.
2416 }
2417
2418 \msg_new:nnnn { cmd } { missing-required }
2419 { Required~argument~missing~for~#1. }
2420 {
2421   The~#1~expects~one~of~its~arguments~to~start~with~'#2'.
2422   LaTeX~did~not~find~this~argument~and~will~insert~a~default~value~for~further~processing.
2423 }
2424

```

```

2425 \msg_new:nnnn { cmd } { non-xparse }
2426   { \str_uppercase:n #1-not-defined-using-xparse. }
2427   {
2428     You~have~asked~for~the~argument~specification~for~the~#1,~
2429     but~this~was~not~defined~using~xparse.
2430   }
2431 \msg_new:nnnn { cmd } { arg-split }
2432   { Too-many-'#1'-separators-in-argument. }
2433   {
2434     LaTeX-was~asked~to~split~the~input~'#3'~
2435     at~each~occurrence~of~the~separator~'#1'~into~#2-parts.~
2436     Too-many~separators~were~found.
2437   }
2438 \msg_new:nnnn { cmd } { unknown }
2439   { Unknown-document-#1. }
2440   {
2441     You~have~asked~for~the~argument~specification~for~the~#1,~
2442     but~it~is~not~defined.
2443   }
2444 \msg_new:nnnn { cmd } { verbatim-nl }
2445   { Verbatim-like-#1-ended-by-end-of-line. }
2446   {
2447     The~verbatim~argument~of~the~#1~cannot~contain~more~than~one~line,~
2448     but~the~end~
2449     of~the~current~line~has~been~reached.~You~may~have~forgotten~the~
2450     closing~delimiter.
2451   \\ \\
2452   LaTeX-will~ignore~'#2'~and~you~may~get~some~additional~
2453     (low-level)~errors.
2454   }
2455 \msg_new:nnnn { cmd } { verbatim-tokenized }
2456   { Verbatim-like-#1-illegal-in-argument. }
2457   {
2458     The~#1-takes~a~verbatim~argument~and~should~therefore~normally~
2459     not~be~used~in~arguments~of~other~commands~or~environments.~
2460     LaTeX-found~an~illegal~token~\tl_if_empty:nF {#3} { (#3)~}
2461     after~'#2'~and~will~drop~everything~up~to~this~point.
2462   \\ \\
2463   Expect-further~(low-level)~errors.
2464 }

Intended more for information.

2465 \msg_new:nnn { cmd } { define-command }      % should be just ``define'' but dep in xparse
2466   {
2467     Defining~command-#1~
2468     with-sig.-'#2'~\msg_line_context:.
2469   }
2470 \msg_new:nnn { cmd } { define-env }
2471   {
2472     Defining~environment-'#1'~
2473     with-sig.-'#2'~\msg_line_context:.
2474   }
2475 \msg_new:nnn { cmd } { redefine }
2476   {
2477     Redefining~command-#1~

```

```

2478     with~sig.~'#2'~\msg_line_context:..
2479   }
2480 \msg_new:nnn { cmd } { redefine-env }
2481 {
2482   Redefining~environment~'#1'~
2483   with~sig.~'#2'~\msg_line_context:..
2484 }
2485 \msg_new:nnn { cmd } { optional-mandatory }
2486 {
2487   Optional~and~mandatory~argument~with~same~delimiter~'#2'.
2488   \\ \\
2489   The~mandatory~argument~specified~with~
2490   '\str_case:nnF{#1}{ {R/r}{r'~or~'R} }{#1}'~has~the~
2491   same~delimiter~'#2'~as~an~earlier~optional~argument.~
2492   It~will~therefore~not~be~possible~to~omit~all~the~earlier~
2493   optional~arguments~when~calling~this~command.
2494   \\ \\
2495   This~may~be~intentional,~but~then~it~might~be~a~mistake.
2496 }
2497 \msg_new:nnn { cmd } { unsupported-let }
2498 {
2499   The~command~'#1'~was~undefined~but~not~the~associated~commands~
2500   '#1~code'~and/or~'#1~defaults'.~Maybe~you~tried~using~
2501   \iow_char:N\\let.~This~may~lead~to~an~infinite~loop.
2502 }

```

1.14 User functions

The user functions are more or less just the internal functions renamed.

\BooleanFalse Design-space names for the Boolean values.

```

2503 \cs_new_eq:NN \BooleanFalse \c_false_bool
2504 \cs_new_eq:NN \BooleanTrue \c_true_bool

```

(End definition for \BooleanFalse and \BooleanTrue.)

\NewDocumentCommand \RenewDocumentCommand \ProvideDocumentCommand \DeclareDocumentCommand The user macros are pretty simple wrappers around the internal ones. There is however a check that the first argument is a single token, possibly surrounded by spaces (hence the strange \use:nnn), and is definable.

```

2505 \cs_new_protected:Npn \NewDocumentCommand #1#2#3
2506 {
2507   \__cmd_check_definable:nNT {#1} \NewDocumentCommand
2508   {
2509     \cs_if_exist:NTF #1
2510     {
2511       \msg_error:nnxx { cmd } { already-defined }
2512       { \use:nnn \token_to_str:N #1 { } }
2513       { \token_to_str:N \NewDocumentCommand }
2514     }
2515     { \__cmd_declare_cmd:Nnn #1 {#2} {#3} }
2516   }
2517 }
2518 \cs_new_protected:Npn \RenewDocumentCommand #1#2#3
2519 {

```

```

2520     \__cmd_check_definable:nNT {#1} \RenewDocumentCommand
2521     {
2522         \cs_if_exist:NTF #1
2523         {
2524             \__cmd_declare_cmd:Nnn #1 {#2} {#3}
2525             \msg_error:nnxx { cmd } { undefined }
2526             { \use:nnn \token_to_str:N #1 { } }
2527             { \token_to_str:N \RenewDocumentCommand }
2528         }
2529     }
2530 }
2531 \cs_new_protected:Npn \ProvideDocumentCommand #1#2#3
2532 {
2533     \__cmd_check_definable:nNT {#1} \ProvideDocumentCommand
2534     { \cs_if_exist:NF #1 { \__cmd_declare_cmd:Nnn #1 {#2} {#3} } }
2535 }
2536 \cs_new_protected:Npn \DeclareDocumentCommand #1#2#3
2537 {
2538     \__cmd_check_definable:nNT {#1} \DeclareDocumentCommand
2539     { \__cmd_declare_cmd:Nnn #1 {#2} {#3} }
2540 }

```

(End definition for `\NewDocumentCommand` and others.)

`\NewDocumentEnvironment` Very similar for environments.

```

\RenewDocumentEnvironment
\ProvideDocumentEnvironment
\DeclareDocumentEnvironment
2541 \cs_new_protected:Npn \NewDocumentEnvironment #1#2#3#4
2542 {
2543     \cs_if_exist:cTF {#1}
2544     { \msg_error:nnx { cmd } { env-already-defined } {#1} }
2545     {
2546         \cs_if_exist:cTF { end #1 }
2547         { \msg_error:nnx { cmd } { env-end-already-defined } {#1} }
2548         { \__cmd_declare_env:nnnn {#1} {#2} {#3} {#4} }
2549     }
2550 }
2551 \cs_new_protected:Npn \RenewDocumentEnvironment #1#2#3#4
2552 {
2553     \cs_if_exist:cTF {#1}
2554     { \__cmd_declare_env:nnnn {#1} {#2} {#3} {#4} }
2555     { \msg_error:nnx { cmd } { env-undefined } {#1} }
2556 }
2557 \cs_new_protected:Npn \ProvideDocumentEnvironment #1#2#3#4
2558 { \cs_if_exist:cF {#1} { \__cmd_declare_env:nnnn {#1} {#2} {#3} {#4} } }
2559 \cs_new_protected:Npn \DeclareDocumentEnvironment #1#2#3#4
2560 { \__cmd_declare_env:nnnn {#1} {#2} {#3} {#4} }


```

(End definition for `\NewDocumentEnvironment` and others.)

`\NewExpandableDocumentCommand`
`\RenewExpandableDocumentCommand`
`\ProvideExpandableDocumentCommand`
`\DeclareExpandableDocumentCommand`

```

2561 \cs_new_protected:Npn \NewExpandableDocumentCommand #1#2#3
2562 {
2563     \__cmd_check_definable:nNT {#1} \NewExpandableDocumentCommand

```

```

2564     {
2565         \cs_if_exist:NTF #1
2566         {
2567             \msg_error:nnxx { cmd } { already-defined }
2568             { \use:n \token_to_str:N #1 { } }
2569             { \token_to_str:N \NewExpandableDocumentCommand }
2570         }
2571         { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
2572     }
2573 }
2574 \cs_new_protected:Npn \RenewExpandableDocumentCommand #1#2#3
2575 {
2576     \__cmd_check_definable:nNT {#1} \RenewExpandableDocumentCommand
2577     {
2578         \cs_if_exist:NTF #1
2579         { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
2580         {
2581             \msg_error:nnxx { cmd } { undefined }
2582             { \use:n \token_to_str:N #1 { } }
2583             { \token_to_str:N \RenewExpandableDocumentCommand }
2584         }
2585     }
2586 }
2587 \cs_new_protected:Npn \ProvideExpandableDocumentCommand #1#2#3
2588 {
2589     \__cmd_check_definable:nNT {#1} \ProvideExpandableDocumentCommand
2590     {
2591         \cs_if_exist:NF #1
2592         { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
2593     }
2594 }
2595 \cs_new_protected:Npn \DeclareExpandableDocumentCommand #1#2#3
2596 {
2597     \__cmd_check_definable:nNT {#1} \DeclareExpandableDocumentCommand
2598     { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
2599 }

```

(End definition for `\NewExpandableDocumentCommand` and others.)

`\IfBooleanT` The logical `<true>` and `<false>` statements are just the normal `\c_true_bool` and `\c_false_bool` so `\bool_if:NTF` is almost enough. However, this code-level function blows up badly when passed invalid input. We want `\IfBooleanTF` to accept a single (non-space) token equal to `\c_true_bool` or `\c_false_bool`, possibly surrounded by spaces. If the input is blank or multiple items, jump to the error and pick the false branch. If the input, ignoring spaces (we do this by omitting braces in the `\tl_if_single_token:nF` test), is not a single token then jump to the error as well. It is then safe to compare the token to the two booleans, picking the appropriate branch. If neither matches, we jump to the error as well.

```

2600 \cs_new:Npn \IfBooleanTF #1
2601 {
2602     \tl_if_single:nF {#1}
2603     { \prg_break:n { \use:n } }
2604     \tl_if_single_token:nF #1
2605     { \prg_break:n { \use:n } }

```

```

2606   \token_if_eq_meaning:NNT #1 \c_true_bool
2607   { \prg_break:n { \use_ii:nnn } }
2608   \token_if_eq_meaning:NNT #1 \c_false_bool
2609   { \prg_break:n { \use_iii:nnn } }
2610   \prg_break:n { \use:n }
2611   \prg_break_point:
2612   {
2613     \msg_expandable_error:nnn { cmd } { if-boolean } {#1}
2614     \use_ii:nn
2615   }
2616 }
2617 \cs_new:Npn \IfBooleanT #1#2 { \IfBooleanTF {#1} {#2} { } }
2618 \cs_new:Npn \IfBooleanF #1 { \IfBooleanTF {#1} { } }

(End definition for \IfBooleanT, \IfBooleanF, and \IfBooleanTF.)

```

\IfNoValueT Simple re-naming.

```

\IfNoValueF 2619 \cs_new_eq:NN \IfNoValueF \tl_if_novalue:nF
\IfNoValueTF 2620 \cs_new_eq:NN \IfNoValueT \tl_if_novalue:nT
2621 \cs_new_eq:NN \IfNoValueTF \tl_if_novalue:nTF

```

(End definition for \IfNoValueT, \IfNoValueF, and \IfNoValueTF.)

\IfValueT Inverted logic.

```

\IfValueF 2622 \cs_new:Npn \IfValueF { \tl_if_novalue:nT }
\IfValueTF 2623 \cs_new:Npn \IfValueT { \tl_if_novalue:nF }
2624 \cs_new:Npn \IfValueTF #1#2#3 { \tl_if_novalue:nTF {#1} {#3} {#2} }

(End definition for \IfValueT, \IfValueF, and \IfValueTF.)

```

\IfBlankT Another simple re-naming.

```

\IfBlankF 2625 <|latexrelease> \IncludeInRelease{2022/06/01}%
\IfBlankTF 2626 <|latexrelease> \IfBlankTF{Testing~for~empty~or~blank}%
2627 \cs_new_eq:NN \IfBlankF \tl_if_blank:nF
2628 \cs_new_eq:NN \IfBlankT \tl_if_blank:nT
2629 \cs_new_eq:NN \IfBlankTF \tl_if_blank:nTF
2630 <|latexrelease> \EndIncludeInRelease
2631 <|latexrelease> \IncludeInRelease{2021/11/15}%
2632 <|latexrelease> \IfBlankTF{Testing~for~empty~or~blank}%
2633 <|latexrelease> \cs_undefine:N \IfBlankF
2634 <|latexrelease> \cs_undefine:N \IfBlankT
2635 <|latexrelease> \cs_undefine:N \IfBlankTF
2636 <|latexrelease>
2637 <|latexrelease> \EndIncludeInRelease

```

(End definition for \IfBlankT, \IfBlankF, and \IfBlankTF.)

\ProcessedArgument Processed arguments are returned using this name, which is reserved here although the definition will change.

```
2638 \tl_new:N \ProcessedArgument
```

(End definition for \ProcessedArgument.)

```

\ReverseBoolean Simple copies.
\SplitArgument 2639 \cs_new_eq:NN \ReverseBoolean \__cmd_bool_reverse:N
  \SplitList 2640 \cs_new_eq:NN \SplitArgument \__cmd_split_argument:nnn
\TrimSpaces 2641 \cs_new_eq:NN \SplitList \__cmd_split_list:nn
  2642 \cs_new_eq:NN \TrimSpaces \__cmd_trim_spaces:n

(End definition for \ReverseBoolean and others.)

\ProcessList To support \SplitList.
  2643 \cs_new_eq:NN \ProcessList \tl_map_function:nN

(End definition for \ProcessList.)

\nCommandArgSpec More simple mappings, with a check that the argument is a single control sequence or
\cumentEnvironmentArgSpec active character.
\ntCommandArgSpec 2644 \cs_new_protected:Npn \GetDocumentCommandArgSpec #1
\cumentEnvironmentArgSpec 2645 {
  \__cmd_check_definable:nNT {#1} \GetDocumentCommandArgSpec
  2646 { \__cmd_get_arg_spec:N #1 }
  2647 }

\ntCommandArgSpec 2648 \cs_new_eq:NN \GetDocumentEnvironmentArgSpec \__cmd_get_arg_spec:n
\cumentEnvironmentArgSpec 2649 \cs_new_protected:Npn \ShowDocumentCommandArgSpec #1
  2650 {
  \__cmd_check_definable:nNT {#1} \ShowDocumentCommandArgSpec
  2651 { \__cmd_show_arg_spec:N #1 }
  2652 }

\ntCommandArgSpec 2653 \cs_new_eq:NN \ShowDocumentEnvironmentArgSpec \__cmd_show_arg_spec:n
  2654 }

\ntCommandArgSpec 2655 \cs_new_eq:NN \ShowDocumentEnvironmentArgSpec \__cmd_show_arg_spec:n

(End definition for \GetDocumentCommandArgSpec and others.)

Finally as promised, restore \__kernel_chk_if_free_cs:N:
\ntCommandArgSpec 2656 \\cs_gset_eq:NN \__kernel_chk_if_free_cs:N \__cmd_chk_if_free_cs:N
\cumentEnvironmentArgSpec 2657 \\cs_undefine:N \__cmd_chk_if_free_cs:N

\ntCommandArgSpec 2658 \
\cumentEnvironmentArgSpec 2659 \\IncludeInRelease{0000/00/00}{\ltcmd}{%
  2660 \ \Document~command~parser}%
\cumentEnvironmentArgSpec 2661 \
\ntCommandArgSpec 2662 \\EndModuleRelease
\cumentEnvironmentArgSpec 2663 \ExplSyntaxOff

Now in \textrm{\textrm{latexrelease}} mode, redefine \NewDocumentCommand to not complain on
commands already defined.

\ntCommandArgSpec 2664 \\IfUndefined{\ExplSyntaxOff}{}{\@postltcmd}
\cumentEnvironmentArgSpec 2665 \\catcode`\\^@=\@catcode@null\relax
\ntCommandArgSpec 2666 \</2ekernel | \textrm{\textrm{latexrelease}}>

We need to stop DocStrip treating @@ in a special way at this point.

\ntCommandArgSpec 2667 \<@@=>

```

File h

lthooks.dtx

Contents

1 Introduction

Hooks are points in the code of commands or environments where it is possible to add processing code into existing commands. This can be done by different packages that do not know about each other and to allow for hopefully safe processing it is necessary to sort different chunks of code added by different packages into a suitable processing order.

This is done by the packages adding chunks of code (via `\AddToHook`) and labeling their code with some label by default using the package name as a label.

At `\begin{document}` all code for a hook is then sorted according to some rules (given by `\DeclareHookRule`) for fast execution without processing overhead. If the hook code is modified afterwards (or the rules are changed), a new version for fast processing is generated.

Some hooks are used already in the preamble of the document. If that happens then the hook is prepared for execution (and sorted) already at that point.

2 Package writer interface

The hook management system is offered as a set of CamelCase commands for traditional L^AT_EX 2 _{ε} packages (and for use in the document preamble if needed) as well as `expl3` commands for modern packages, that use the L3 programming layer of L^AT_EX. Behind the scenes, a single set of data structures is accessed so that packages from both worlds can coexist and access hooks in other packages.

2.1 L^AT_EX 2 _{ε} interfaces

2.1.1 Declaring hooks

With a few exceptions, hooks have to be declared before they can be used. The exceptions are the generic hooks for commands and environments (executed at `\begin` and `\end`), and the hooks run when loading files (see section 3.1).

`\NewHook \NewHook {⟨hook⟩}`

Creates a new `⟨hook⟩`. If this hook is declared within a package it is suggested that its name is always structured as follows: `⟨package-name⟩/⟨hook-name⟩`. If necessary you can further subdivide the name by adding more / parts. If a hook name is already taken, an error is raised and the hook is not created.

The `⟨hook⟩` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

```
\NewReversedHook \NewReversedHook {\<hook>}
```

Like `\NewHook` declares a new `\<hook>`. the difference is that the code chunks for this hook are in reverse order by default (those added last are executed first). Any rules for the hook are applied after the default ordering. See sections 2.3 and 2.4 for further details.

The `\<hook>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

```
\NewMirroredHookPair \NewMirroredHookPair {\<hook-1>} {\<hook-2>}
```

A shorthand for `\NewHook{\<hook-1>} \NewReversedHook{\<hook-2>}`.

The `\<hooks>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

2.1.2 Special declarations for generic hooks

The declarations here should normally not be used. They are available to provide support for special use cases mainly involving generic command hooks.

```
\DisableGenericHook \DisableGenericHook {\<hook>}
```

After this declaration³ the `\<hook>` is no longer usable: Any attempt to add further code to it will result in an error and any use, e.g., via `\UseHook`, will simply do nothing.

This is intended to be used with generic command hooks (see `ltcmdhooks-doc`) as depending on the definition of the command such generic hooks may be unusable. If that is known, a package developer can disable such hooks up front.

The `\<hook>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

```
\ActivateGenericHook \ActivateGenericHook {\<hook>}
```

This declaration activates a generic hook provided by a package/class (e.g., one used in code with `\UseHook` or `\UseOneTimeHook`) without it being explicitly declared with `\NewHook`). This command undoes the effect of `\DisableGenericHook`. If the hook is already activated, this command does nothing.

See section 2.6 for a discussion of when this declaration is appropriate.

2.1.3 Using hooks in code

```
\UseHook \UseHook {\<hook>}
```

Execute the hook code inside a command or environment.

Before `\begin{document}` the fast execution code for a hook is not set up, so in order to use a hook there it is explicitly initialized first. As that involves assignments using a hook at those times is not 100% the same as using it after `\begin{document}`.

The `\<hook>` cannot be specified using the dot-syntax. A leading `.` is treated literally.

³In the 2020/06 release this command was called `\DisableHook`, but that name was misleading as it shouldn't be used to disable non-generic hooks.

```
\UseOneTimeHook \UseOneTimeHook {\<hook>}
```

Some hooks are only used (and can be only used) in one place, for example, those in `\begin{document}` or `\end{document}`. Once we have passed that point adding to the hook through a defined `\<addto-cmd>` command (e.g., `\AddToHook` or `\AtBeginDocument`, etc.) would have no effect (as would the use of such a command inside the hook code itself). It is therefore customary to redefine `\<addto-cmd>` to simply process its argument, i.e., essentially make it behave like `\@firstofone`.

`\UseOneTimeHook` does that: it records that the hook has been consumed and any further attempt to add to it will result in executing the code to be added immediately.

The `\<hook>` cannot be specified using the dot-syntax. A leading `.` is treated literally. See section 2.1.5 for details.

Using `\UseOneTimeHook` several times with the same `\{\<hook>\}` means that it only executes the first time it is used. For example, if it is used in a command that can be called several times then the hook executes during only the *first* invocation of that command; this allows its use as an “initialization hook”.

Mixing `\UseHook` and `\UseOneTimeHook` for the same `\{\<hook>\}` should be avoided, but if this is done then neither will execute after the first `\UseOneTimeHook`.

2.1.4 Updating code for hooks

```
\AddToHook \AddToHook {\<hook>}[{\<label>}]{\<code>}
```

Adds `\<code>` to the `\<hook>` labeled by `\<label>`. When the optional argument `\<label>` is not provided, the `\<default label>` is used (see section 2.1.5). If `\AddToHook` is used in a package/class, the `\<default label>` is the package/class name, otherwise it is `top-level` (the `top-level` label is treated differently: see section 2.1.6).

If there already exists code under the `\<label>` then the new `\<code>` is appended to the existing one (even if this is a reversed hook). If you want to replace existing code under the `\<label>`, first apply `\RemoveFromHook`.

The hook doesn’t have to exist for code to be added to it. However, if it is not declared, then obviously the added `\<code>` will never be executed. This allows for hooks to work regardless of package loading order and enables packages to add to hooks from other packages without worrying whether they are actually used in the current document. See section 2.1.8.

The `\<hook>` and `\<label>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

```
\RemoveFromHook \RemoveFromHook {\hook}{[\label]}
```

Removes any code labeled by *label* from the *hook*. When the optional argument *label* is not provided, the *default label* is used (see section 2.1.5).

If there is no code under the *label* in the *hook*, or if the *hook* does not exist, a warning is issued when you attempt to \RemoveFromHook, and the command is ignored. \RemoveFromHook should be used only when you know exactly what labels are in a hook. Typically this will be when some code gets added to a hook by a package, then later this code is removed by that same package. If you want to prevent the execution of code from another package, use the **voids** rule instead (see section 2.1.7).

If the optional *label* argument is *, then all code chunks are removed. This is rather dangerous as it may well drop code from other packages (that one may not know about); it should therefore not be used in packages but only in document preambles!

The *hook* and *label* can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

In contrast to the **voids** relationship between two labels in a \DeclareHookRule this is a destructive operation as the labeled code is removed from the hook data structure, whereas the relationship setting can be undone by providing a different relationship later.

A useful application for this declaration inside the document body is when one wants to temporarily add code to hooks and later remove it again, e.g.,

```
\AddToHook{env/quote/before}{\small}
\begin{quote}
  A quote set in a smaller typeface
\end{quote}
...
\RemoveFromHook{env/quote/before}
... now back to normal for further quotes
```

Note that you can't cancel the setting with

```
\AddToHook{env/quote/before}{}  
because that only "adds" a further empty chunk of code to the hook. Adding  
\normalsize would work but that means the hook then contained \small\normalsize  
which means two font size changes for no good reason.
```

The above is only needed if one wants to typeset several quotes in a smaller typeface. If the hook is only needed once then \AddToHookNext is simpler, because it resets itself after one use.

```
\AddToHookNext \AddToHookNext {\hook}{\code}
```

Adds $\langle code \rangle$ to the next invocation of the $\langle hook \rangle$. The code is executed after the normal hook code has finished and it is executed only once, i.e. it is deleted after it was used.

Using this declaration is a global operation, i.e., the code is not lost even if the declaration is used inside a group and the next invocation of the hook happens after the end of that group. If the declaration is used several times before the hook is executed then all code is executed in the order in which it was declared.⁴

If this declaration is used with a one-time hook then the code is only ever used if the declaration comes before the hook's invocation. This is because, in contrast to `\AddToHook`, the code in this declaration is not executed immediately in the case when the invocation of the hook has already happened—in other words, this code will truly execute only on the next invocation of the hook (and in the case of a one-time hook there is no such “next invocation”). This gives you a choice: should my code execute always, or should it execute only at the point where the one-time hook is used (and not at all if this is impossible)? For both of these possibilities there are use cases.

It is possible to nest this declaration using the same hook (or different hooks): e.g.,

```
\AddToHookNext{\hook}{\code-1}\AddToHookNext{\hook}{\code-2}}
```

will execute $\langle code-1 \rangle$ next time the $\langle hook \rangle$ is used and at that point puts $\langle code-2 \rangle$ into the $\langle hook \rangle$ so that it gets executed on following time the hook is run.

A hook doesn't have to exist for code to be added to it. This allows for hooks to work regardless of package loading order. See section 2.1.8.

The $\langle hook \rangle$ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

```
\ClearHookNext \ClearHookNext{\hook}
```

Normally `\AddToHookNext` is only used when you know precisely where it will apply and why you want some extra code at that point. However, there are a few use cases in which such a declaration needs to be canceled, for example, when discarding a page with `\DiscardShipoutBox` (but even then not always), and in such situations `\ClearHookNext` can be used.

2.1.5 Hook names and default labels

It is best practice to use `\AddToHook` in packages or classes *without specifying a $\langle label \rangle$* because then the package or class name is automatically used, which is helpful if rules are needed, and avoids mistyping the $\langle label \rangle$.

Using an explicit $\langle label \rangle$ is only necessary in very specific situations, e.g., if you want to add several chunks of code into a single hook and have them placed in different parts of the hook (by providing some rules).

The other case is when you develop a larger package with several sub-packages. In that case you may want to use the same $\langle label \rangle$ throughout the sub-packages in order to avoid that the labels change if you internally reorganize your code.

Except for `\UseHook`, `\UseOneTimeHook` and `\IfHookEmptyTF` (and their `\Expl3` interfaces `\hook_use:n`, `\hook_use_once:n` and `\hook_if_empty:nTF`), all $\langle hook \rangle$ and $\langle label \rangle$ arguments are processed in the same way: first, spaces are trimmed around the argument, then it is fully expanded until only character tokens remain. If the full expansion

⁴There is no mechanism to reorder such code chunks (or delete them).

of the $\langle hook \rangle$ or $\langle label \rangle$ contains a non-expandable non-character token, a low-level TeX error is raised (namely, the $\langle hook \rangle$ is expanded using TeX's `\csname... \endcsname`, as such, Unicode characters are allowed in $\langle hook \rangle$ and $\langle label \rangle$ arguments). The arguments of `\UseHook`, `\UseOneTimeHook`, and `\IfHookEmptyTF` are processed much in the same way except that spaces are not trimmed around the argument, for better performance.

It is not enforced, but highly recommended that the hooks defined by a package, and the $\langle labels \rangle$ used to add code to other hooks contain the package name to easily identify the source of the code chunk and to prevent clashes. This should be the standard practice, so this hook management code provides a shortcut to refer to the current package in the name of a $\langle hook \rangle$ and in a $\langle label \rangle$. If the $\langle hook \rangle$ name or the $\langle label \rangle$ consist just of a single dot (.), or starts with a dot followed by a slash (/) then the dot denotes the $\langle default label \rangle$ (usually the current package or class name—see `\SetDefaultHookLabel`). A “.” or “./” anywhere else in a $\langle hook \rangle$ or in $\langle label \rangle$ is treated literally and is not replaced.

For example, inside the package `mypackage.sty`, the default label is `mypackage`, so the instructions:

```
\NewHook  {./hook}
\AddToHook {./hook}[]{}{code}      % Same as \AddToHook{./hook}{code}
\AddToHook {./hook}[/sub]{}{code}
\DeclareHookRule{begindocument}{}{before}{babel}
\AddToHook {file/foo.tex/after}{code}
```

are equivalent to:

```
\NewHook  {mymodule/hook}
\AddToHook {mymodule/hook}[mymodule]{code}
\AddToHook {mymodule/hook}[mymodule/sub]{code}
\DeclareHookRule{begindocument}{mymodule}{before}{babel}
\AddToHook {file/foo.tex/after}{code}           % unchanged
```

The $\langle default label \rangle$ is automatically set equal to the name of the current package or class at the time the package is loaded. If the hook command is used outside of a package, or the current file wasn't loaded with `\usepackage` or `\documentclass`, then the `top-level` is used as the $\langle default label \rangle$. This may have exceptions—see `\PushDefaultHookLabel`.

This syntax is available in all $\langle label \rangle$ arguments and most $\langle hook \rangle$ arguments, both in the L^AT_EX 2 _{ϵ} interface, and the L^AT_EX3 interface described in section 2.2.

Note, however, that the replacement of . by the $\langle default label \rangle$ takes place when the hook command is executed, so actions that are somehow executed after the package ends will have the wrong $\langle default label \rangle$ if the dot-syntax is used. For that reason, this syntax is not available in `\UseHook` (and `\hook_use:n`) because the hook is most of the time used outside of the package file in which it was defined. This syntax is also not available in the hook conditionals `\IfHookEmptyTF` (and `\hook_if_empty:nTF`), because these conditionals are used in some performance-critical parts of the hook management code, and because they are usually used to refer to other package's hooks, so the dot-syntax doesn't make much sense.

In some cases, for example in large packages, one may want to separate it in logical parts, but still use the main package name as $\langle label \rangle$, then the $\langle default label \rangle$ can be set using `\SetDefaultHookLabel` or `\PushDefaultHookLabel{...}... \PopDefaultHookLabel`.

```
\PushDefaultHookLabel \PushDefaultHookLabel {\<default label>}
\PopDefaultHookLabel   <code>
\PopDefaultHookLabel
```

\PushDefaultHookLabel sets the current *<default label>* to be used in *<label>* arguments, or when replacing a leading “.” (see above). \PopDefaultHookLabel reverts the *<default label>* to its previous value.

Inside a package or class, the *<default label>* is equal to the package or class name, unless explicitly changed. Everywhere else, the *<default label>* is **top-level** (see section 2.1.6) unless explicitly changed.

The effect of \PushDefaultHookLabel holds until the next \PopDefaultHookLabel. \usepackage (and \RequirePackage and \documentclass) internally use

```
\PushDefaultHookLabel{\<package name>}
  <package code>
\PopDefaultHookLabel
```

to set the *<default label>* for the package or class file. Inside the *<package code>* the *<default label>* can also be changed with \SetDefaultHookLabel. \input and other file input-related commands from the L^AT_EX kernel do not use \PushDefaultHookLabel, so code within files loaded by these commands does *not* get a dedicated *<label>*! (that is, the *<default label>* is the current active one when the file was loaded.)

Packages that provide their own package-like interfaces (TikZ’s \usetikzlibrary, for example) can use \PushDefaultHookLabel and \PopDefaultHookLabel to set dedicated labels and to emulate \usepackage-like hook behavior within those contexts.

The **top-level** label is treated differently, and is reserved to the user document, so it is not allowed to change the *<default label>* to **top-level**.

```
\SetDefaultHookLabel \SetDefaultHookLabel {\<default label>}
```

Similarly to \PushDefaultHookLabel, sets the current *<default label>* to be used in *<label>* arguments, or when replacing a leading “.”. The effect holds until the label is changed again or until the next \PopDefaultHookLabel. The difference between \PushDefaultHookLabel and \SetDefaultHookLabel is that the latter does not save the current *<default label>*.

This command is useful when a large package is composed of several smaller packages, but all should have the same *<label>*, so \SetDefaultHookLabel can be used at the beginning of each package file to set the correct label.

\SetDefaultHookLabel is not allowed in the main document, where the *<default label>* is **top-level** and there is no \PopDefaultHookLabel to end its effect. It is also not allowed to change the *<default label>* to **top-level**.

2.1.6 The top-level label

The **top-level** label, assigned to code added from the main document, is different from other labels. Code added to hooks (usually \AtBeginDocument) in the preamble is almost always to change something defined by a package, so it should go at the very end of the hook.

Therefore, code added in the **top-level** is always executed at the end of the hook, regardless of where it was declared. If the hook is reversed (see \NewReversedHook), the **top-level** chunk is executed at the very beginning instead.

Rules regarding `top-level` have no effect: if a user wants to have a specific set of rules for a code chunk, they should use a different label to said code chunk, and provide a rule for that label instead.

The `top-level` label is exclusive for the user, so trying to add code with that label from a package results in an error.

2.1.7 Defining relations between hook code

The default assumption is that code added to hooks by different packages are independent and the order in which they are executed is irrelevant. While this is true in many cases it is obviously false in others.

Before the hook management system was introduced packages had to take elaborate precaution to determine of some other package got loaded as well (before or after) and find some ways to alter its behavior accordingly. In addition it was often the user's responsibility to load packages in the right order so that code added to hooks got added in the right order and some cases even altering the loading order wouldn't resolve the conflicts.

With the new hook management system it is now possible to define rules (i.e., relationships) between code chunks added by different packages and explicitly describe in which order they should be processed.

```
\DeclareHookRule \DeclareHookRule {\langle hook\rangle}{\langle label1\rangle}{\langle relation\rangle}{\langle label2\rangle}
```

Defines a relation between $\langle label1 \rangle$ and $\langle label2 \rangle$ for a given $\langle hook \rangle$. If $\langle hook \rangle$ is `??` this defines a default relation for all hooks that use the two labels, i.e., that have chunks of code labeled with $\langle label1 \rangle$ and $\langle label2 \rangle$. Rules specific to a given hook take precedence over default rules that use `??` as the $\langle hook \rangle$.

Currently, the supported relations are the following:

`before` or `<` Code for $\langle label1 \rangle$ comes before code for $\langle label2 \rangle$.

`after` or `>` Code for $\langle label1 \rangle$ comes after code for $\langle label2 \rangle$.

`incompatible-warning` Only code for either $\langle label1 \rangle$ or $\langle label2 \rangle$ can appear for that hook (a way to say that two packages—or parts of them—are incompatible). A warning is raised if both labels appear in the same hook.

`incompatible-error` Like `incompatible-warning` but instead of a warning a L^AT_EX error is raised, and the code for both labels are dropped from that hook until the conflict is resolved.

`voids` Code for $\langle label1 \rangle$ overwrites code for $\langle label2 \rangle$. More precisely, code for $\langle label2 \rangle$ is dropped for that hook. This can be used, for example if one package is a superset in functionality of another one and therefore wants to undo code in some hook and replace it with its own version.

`unrelated` The order of code for $\langle label1 \rangle$ and $\langle label2 \rangle$ is irrelevant. This rule is there to undo an incorrect rule specified earlier.

There can only be a single relation between two labels for a given hook, i.e., a later `\DeclareHookrule` overwrites any previous declaration.

The $\langle hook \rangle$ and $\langle label \rangle$ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

```
\ClearHookRule \ClearHookRule{\hook}{\label1}{\label2}
```

Syntactic sugar for saying that $\langle\text{label1}\rangle$ and $\langle\text{label2}\rangle$ are unrelated for the given $\langle\text{hook}\rangle$.

```
\DeclareDefaultHookRule \DeclareDefaultHookRule{\label1}{\relation}{\label2}
```

This sets up a relation between $\langle\text{label1}\rangle$ and $\langle\text{label2}\rangle$ for all hooks unless overwritten by a specific rule for a hook. Useful for cases where one package has a specific relation to some other package, e.g., is `incompatible` or always needs a special ordering `before` or `after`. (Technically it is just a shorthand for using `\DeclareHookRule` with `??` as the hook name.)

Declaring default rules is only supported in the document preamble.⁵

The $\langle\text{label}\rangle$ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

2.1.8 Querying hooks

Simpler data types, like token lists, have three possible states; they can:

- exist and be empty;
- exist and be non-empty; and
- not exist (in which case emptiness doesn't apply);

Hooks are a bit more complicated: a hook may exist or not, and independently it may or may not be empty. This means that even a hook that doesn't exist may be non-empty and it can also be disabled.

This seemingly strange state may happen when, for example, package *A* defines hook *A/foo*, and package *B* adds some code to that hook. However, a document may load package *B* before package *A*, or may not load package *A* at all. In both cases some code is added to hook *A/foo* without that hook being defined yet, thus that hook is said to be non-empty, whereas it doesn't exist. Therefore, querying the existence of a hook doesn't imply its emptiness, neither does the other way around.

Given that code or rules can be added to a hook even if it doesn't physically exist yet, means that a querying its existence has no real use case (in contrast to other variables that can only be update if they have already been declared). For that reason only the test for emptiness has a public interface.

A hook is said to be empty when no code was added to it, either to its permanent code pool, or to its “next” token list. The hook doesn't need to be declared to have code added to its code pool. A hook is said to exist when it was declared with `\NewHook` or some variant thereof. Generic hooks such as `file` and `env` hooks are automatically declared when code is added to them.

```
\IfHookEmptyTF * \IfHookEmptyTF {\hook} {\truecode} {\falsecode}
```

Tests if the $\langle\text{hook}\rangle$ is empty (*i.e.*, no code was added to it using either `\AddToHook` or `\AddToHookNext`) or such code was removed again (via `\RemoveFromHook`), and branches to either $\langle\text{true code}\rangle$ or $\langle\text{false code}\rangle$ depending on the result.

The $\langle\text{hook}\rangle$ cannot be specified using the dot-syntax. A leading `.` is treated literally.

⁵Trying to do so, e.g., via `\DeclareHookRule` with `??` has bad side-effects and is not supported (though not explicitly caught for performance reasons).

2.1.9 Displaying hook code

If one has to adjust the code execution in a hook using a hook rule it is helpful to get some information about the code associated with a hook, its current order and the existing rules.

```
\ShowHook \ShowHook {\hook}
\LogHook \LogHook {\hook}
```

Displays information about the *hook* such as

- the code chunks (and their labels) added to it,
- any rules set up to order them,
- the computed order in which the chunks are executed,
- any code executed on the next invocation only.

\LogHook prints the information to the .log file, and \ShowHook prints them to the terminal/command window and starts TeX's prompt (only in \errorstopmode) to wait for user action.

The *hook* can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

Suppose a hook `example-hook` whose output of \ShowHook{example-hook} is:

```
1   -> The hook 'example-hook':
2   > Code chunks:
3   >     foo -> [code from package 'foo']
4   >     bar -> [from package 'bar']
5   >     baz -> [package 'baz' is here]
6   > Document-level (top-level) code (executed last):
7   >     -> [code from 'top-level']
8   > Extra code for next invocation:
9   >     -> [one-time code]
10  > Rules:
11  >     foo|baz with relation >
12  >     baz|bar with default relation <
13  > Execution order (after applying rules):
14  >     baz, foo, bar.
```

In the listing above, lines 3 to 5 show the three code chunks added to the hook and their respective labels in the format

label -> *code*

Line 7 shows the code chunk added by the user in the main document (labeled `top-level`) in the format

```
Document-level (top-level) code (executed {first/last}):
-> (top-level code)
```

This code will be either the first or last code executed by the hook (`last` if the hook is normal, `first` if it is reversed). This chunk is not affected by rules and does not take part in sorting.

Line 9 shows the code chunk for the next execution of the hook in the format

$\rightarrow \langle next\text{-}code \rangle$

This code will be used and disappear at the next `\UseHook{example-hook}`, in contrast to the chunks mentioned earlier, which can only be removed from that hook by doing `\RemoveFromHook{\langle label \rangle}{example-hook}`.

Lines 11 and 12 show the rules declared that affect this hook in the format

$\langle label\text{-}1 \rangle | \langle label\text{-}2 \rangle \text{ with } \langle default? \rangle \text{ relation } \langle relation \rangle$

which means that the $\langle relation \rangle$ applies to $\langle label\text{-}1 \rangle$ and $\langle label\text{-}2 \rangle$, in that order, as detailed in `\DeclareHookRule`. If the relation is `default` it means that this rule applies to $\langle label\text{-}1 \rangle$ and $\langle label\text{-}2 \rangle$ in *all* hooks, (unless overridden by a non-default relation).

Finally, line 14 lists the labels in the hook after sorting; that is, in the order they will be executed when the hook is used.

2.1.10 Debugging hook code

`\DebugHooksOn` `\DebugHooksOn`

`\DebugHooksOff` Turn the debugging of hook code on or off. This displays most changes made to the hook data structures. The output is rather coarse and not really intended for normal use.

2.2 L3 programming layer (`expl3`) interfaces

This is a quick summary of the L^AT_EX3 programming interfaces for use with packages written in `expl3`. In contrast to the L^AT_EX 2_E interfaces they always use mandatory arguments only, e.g., you always have to specify the $\langle label \rangle$ for a code chunk. We therefore suggest to use the declarations discussed in the previous section even in `expl3` packages, but the choice is yours.

<code>\hook_new:n</code>	<code>\hook_new:n {\langle hook \rangle}</code>
<code>\hook_new_reversed:n</code>	<code>\hook_new_reversed:n {\langle hook \rangle}</code>
<code>\hook_new_pair:nn</code>	<code>\hook_new_pair:nn {\langle hook-1 \rangle} {\langle hook-2 \rangle}</code>

Creates a new $\langle hook \rangle$ with normal or reverse ordering of code chunks. `\hook_new_pair:nn` creates a pair of such hooks with $\{\langle hook-2 \rangle\}$ being a reversed hook. If a hook name is already taken, an error is raised and the hook is not created.

The $\langle hook \rangle$ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\hook_disable_generic:n` `\hook_disable_generic:n {\langle hook \rangle}`

Marks $\{\langle hook \rangle\}$ as disabled. Any further attempt to add code to it or declare it, will result in an error and any call to `\hook_use:n` will simply do nothing.

This declaration is intended for use with generic hooks that are known not to work (see `lthcmdhooks-doc`) if they receive code.

The $\langle hook \rangle$ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\hook_activate_generic:n \hook_activate_generic:n {<hook>}`

This is like `\hook_new:n` but it does nothing if the hook was previously declared with `\hook_new:n`. This declaration should be used only in special situations, e.g., when a command from another package needs to be altered and it is not clear whether a generic cmd hook (for that command) has been previously explicitly declared.

Normally `\hook_new:n` should be used instead of this.

`\hook_use:n \hook_use:n {<hook>}`

Executes the `{<hook>}` code followed (if set up) by the code for next invocation only, then empties that next invocation code.

The `<hook>` cannot be specified using the dot-syntax. A leading `.` is treated literally.

`\hook_use_once:n \hook_use_once:n {<hook>}`

Changes the `{<hook>}` status so that from now on any addition to the hook code is executed immediately. Then execute any `{<hook>}` code already set up.

The `<hook>` cannot be specified using the dot-syntax. A leading `.` is treated literally.

`\hook_gput_code:nnn \hook_gput_code:nnn {<hook>} {<label>} {<code>}`

Adds a chunk of `<code>` to the `<hook>` labeled `<label>`. If the label already exists the `<code>` is appended to the already existing code.

If code is added to an external `<hook>` (of the kernel or another package) then the convention is to use the package name as the `<label>` not some internal module name or some other arbitrary string.

The `<hook>` and `<label>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\hook_gput_next_code:nn \hook_gput_next_code:nn {<hook>} {<code>}`

Adds a chunk of `<code>` for use only in the next invocation of the `<hook>`. Once used it is gone.

This is simpler than `\hook_gput_code:nnn`, the code is simply appended to the hook in the order of declaration at the very end, i.e., after all standard code for the hook got executed.

Thus if one needs to undo what the standard does one has to do that as part of `<code>`.

The `<hook>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\hook_gclear_next_code:n \hook_gclear_next_code:n {<hook>}`

Undo any earlier `\hook_gput_next_code:nn`.

`\hook_gremove_code:nn \hook_gremove_code:nn {⟨hook⟩} {⟨label⟩}`

Removes any code for *⟨hook⟩* labeled *⟨label⟩*.

If there is no code under the *⟨label⟩* in the *⟨hook⟩*, or if the *⟨hook⟩* does not exist, a warning is issued when you attempt to use `\hook_gremove_code:nn`, and the command is ignored.

If the second argument is `*`, then all code chunks are removed. This is rather dangerous as it drops code from other packages one may not know about, so think twice before using that!

The *⟨hook⟩* and *⟨label⟩* can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\hook_gset_rule:nnnn \hook_gset_rule:nnnn {⟨hook⟩} {⟨label1⟩} {⟨relation⟩} {⟨label2⟩}`

Relate *⟨label1⟩* with *⟨label2⟩* when used in *⟨hook⟩*. See `\DeclareHookRule` for the allowed *⟨relation⟩*s. If *⟨hook⟩* is `??` a default rule is specified.

The *⟨hook⟩* and *⟨label⟩* can be specified using the dot-syntax to denote the current package name. See section 2.1.5. The dot-syntax is parsed in both *⟨label⟩* arguments, but it usually makes sense to be used in only one of them.

`\hook_if_empty_p:n ∗ \hook_if_empty:nTF {⟨hook⟩} {⟨true code⟩} {⟨false code⟩}`

`\hook_if_empty:nTF ∗` Tests if the *⟨hook⟩* is empty (*i.e.*, no code was added to it using either `\AddToHook` or `\AddToHookNext`), and branches to either *⟨true code⟩* or *⟨false code⟩* depending on the result.

The *⟨hook⟩* cannot be specified using the dot-syntax. A leading `.` is treated literally.

`\hook_show:n \hook_show:n {⟨hook⟩}`
`\hook_log:n \hook_log:n {⟨hook⟩}`

Displays information about the *⟨hook⟩* such as

- the code chunks (and their labels) added to it,
- any rules set up to order them,
- the computed order in which the chunks are executed,
- any code executed on the next invocation only.

`\hook_log:n` prints the information to the `.log` file, and `\hook_show:n` prints them to the terminal/command window and starts TeX's prompt (only if `\errorstopmode`) to wait for user action.

The *⟨hook⟩* can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\hook_debug_on: \hook_debug_on:`

`\hook_debug_off:` Turns the debugging of hook code on or off. This displays changes to the hook data.

2.3 On the order of hook code execution

Chunks of code for a *⟨hook⟩* under different labels are supposed to be independent if there are no special rules set up that define a relation between the chunks. This means that you can't make assumptions about the order of execution!

Suppose you have the following declarations:

```
\NewHook{myhook}
\AddToHook{myhook}{packageA}{\typeout{A}}
\AddToHook{myhook}{packageB}{\typeout{B}}
\AddToHook{myhook}{packageC}{\typeout{C}}
```

then executing the hook with \UseHook will produce the typeout A B C in that order. In other words, the execution order is computed to be packageA, packageB, packageC which you can verify with \ShowHook{myhook}:

```
-> The hook 'myhook':
> Code chunks:
>     packageA -> \typeout {A}
>     packageB -> \typeout {B}
>     packageC -> \typeout {C}
> Document-level (top-level) code (executed last):
>     ---
> Extra code for next invocation:
>     ---
> Rules:
>     ---
> Execution order:
>     packageA, packageB, packageC.
```

The reason is that the code chunks are internally saved in a property list and the initial order of such a property list is the order in which key-value pairs got added. However, that is only true if nothing other than adding happens!

Suppose, for example, you want to replace the code chunk for packageA, e.g.,

```
\RemoveFromHook{myhook}{packageA}
\AddToHook{myhook}{packageA}{\typeout{A alt}}
```

then your order becomes packageB, packageC, packageA because the label got removed from the property list and then re-added (at its end).

While that may not be too surprising, the execution order is also sometimes altered if you add a redundant rule, e.g. if you specify

```
\DeclareHookRule{myhook}{packageA}{before}{packageB}
```

instead of the previous lines we get

```
-> The hook 'myhook':
> Code chunks:
>     packageA -> \typeout {A}
>     packageB -> \typeout {B}
>     packageC -> \typeout {C}
> Document-level (top-level) code (executed last):
>     ---
> Extra code for next invocation:
>     ---
> Rules:
>     packageB|packageA with relation >
> Execution order (after applying rules):
>     packageA, packageC, packageB.
```

As you can see the code chunks are still in the same order, but in the execution order for the labels `packageB` and `packageC` have swapped places. The reason is that, with the rule there are two orders that satisfy it, and the algorithm for sorting happened to pick a different one compared to the case without rules (where it doesn't run at all as there is nothing to resolve). Incidentally, if we had instead specified the redundant rule

```
\DeclareHookRule{myhook}{packageB}{before}{packageC}
```

the execution order would not have changed.

In summary: it is not possible to rely on the order of execution unless there are rules that partially or fully define the order (in which you can rely on them being fulfilled).

2.4 The use of “reversed” hooks

You may have wondered why you can declare a “reversed” hook with `\NewReversedHook` and what that does exactly.

In short: the execution order of a reversed hook (without any rules!) is exactly reversed to the order you would have gotten for a hook declared with `\NewHook`.

This is helpful if you have a pair of hooks where you expect to see code added that involves grouping, e.g., starting an environment in the first and closing that environment in the second hook. To give a somewhat contrived example⁶, suppose there is a package adding the following:

```
\AddToHook{env/quote/before}[package-1]{\begin{itshape}}
\AddToHook{env/quote/after} [package-1]{\end{itshape}}
```

As a result, all quotes will be in italics. Now suppose further that another `package-too` makes the quotes also in blue and therefore adds:

```
\usepackage{color}
\AddToHook{env/quote/before}[package-too]{\begin{color}{blue}}
\AddToHook{env/quote/after} [package-too]{\end{color}}
```

Now if the `env/quote/after` hook would be a normal hook we would get the same execution order in both hooks, namely:

`package-1, package-too`

(or vice versa) and as a result, would get:

```
\begin{itshape}\begin{color}{blue} ...
\end{itshape}\end{color}
```

and an error message that `\begin{color}` ended by `\end{itshape}`. With `env/quote/after` declared as a reversed hook the execution order is reversed and so all environments are closed in the correct sequence and `\ShowHook` would give us the following output:

```
-> The hook 'env/quote/after':
> Code chunks:
>     package-1 -> \end {itshape}
>     package-too -> \end {color}
> Document-level (top-level) code (executed first):
```

⁶there are simpler ways to achieve the same effect.

```

>     ---
> Extra code for next invocation:
>     ---
> Rules:
>     ---
> Execution order (after reversal):
>     package-too, package-1.

```

The reversal of the execution order happens before applying any rules, so if you alter the order you will probably have to alter it in both hooks, not just in one, but that depends on the use case.

2.5 Difference between “normal” and “one-time” hooks

When executing a hook a developer has the choice of using either `\UseHook` or `\UseOneTimeHook` (or their `\Expl3` equivalents `\hook_use:n` and `\hook_use_once:n`). This choice affects how `\AddToHook` is handled after the hook has been executed for the first time.

With normal hooks adding code via `\AddToHook` means that the code chunk is added to the hook data structure and then used each time `\UseHook` is called.

With one-time hooks it this is handled slightly differently: After `\UseOneTimeHook` has been called, any further attempts to add code to the hook via `\AddToHook` will simply execute the `\langle code \rangle` immediately.

This has some consequences one needs to be aware of:

- If `\langle code \rangle` is added to a normal hook after the hook was executed and it is never executed again for one or the other reason, then this new `\langle code \rangle` will never be executed.
- In contrast if that happens with a one-time hook the `\langle code \rangle` is executed immediately.

In particular this means that construct such as

```
\AddToHook{myhook}
  { \langle code-1 \rangle \AddToHook{myhook}{\langle code-2 \rangle} \langle code-3 \rangle }
```

works for one-time hooks⁷ (all three code chunks are executed one after another), but it makes little sense with a normal hook, because with a normal hook the first time `\UseHook{myhook}` is executed it would

- execute `\langle code-1 \rangle`,
- then execute `\AddToHook{myhook}{\langle code-2 \rangle}` which adds the code chunk `\langle code-2 \rangle` to the hook for use on the next invocation,
- and finally execute `\langle code-3 \rangle`.

The second time `\UseHook` is called it would execute the above and in addition `\langle code-2 \rangle` as that was added as a code chunk to the hook in the meantime. So each time the hook is used another copy of `\langle code-2 \rangle` is added and so that code chunk is executed $\langle \# \text{ of invocations} \rangle - 1$ times.

⁷This is sometimes used with `\AtBeginDocument` which is why it is supported.

2.6 Generic hooks provided by packages

The hook management system also implements a category of hooks that are called “Generic Hooks”. Normally a hook has to be explicitly declared before it can be used in code. This ensures that different packages are not using the same hook name for unrelated purposes—something that would result in absolute chaos. However, there are a number of “standard” hooks where it is unreasonable to declare them beforehand, e.g., each and every command has (in theory) an associated `before` and `after` hook. In such cases, i.e., for command, environment or file hooks, they can be used simply by adding code to them with `\AddToHook`. For more specialized generic hooks, e.g., those provided by `babel`, you have to additionally enable them with `\ActivateGenericHook` as explained below.

The generic hooks provided by L^AT_EX are those for `cmd`, `env`, `file`, `include` package, and `class`, and all these are available out of the box: you only have to use `\AddToHook` to add code to them, but you don’t have to add `\UseHook` or `\UseOneTimeHook` to your code, because this is already done for you (or, in the case of `cmd` hooks, the command’s code is patched at `\begin{document}`, if necessary).

However, if you want to provide further generic hooks in your own code, the situation is slightly different. To do this you should use `\UseHook` or `\UseOneTimeHook`, but *without declaring the hook* with `\NewHook`. As mentioned earlier, a call to `\UseHook` with an undeclared hook name does nothing. So as an additional setup step, you need to explicitly activate your generic hook. Note that a generic hook produced in this way is always a normal hook.

For a truly generic hook, with a variable part in the hook name, such upfront activation would be difficult or impossible, because you typically do not know what kind of variable parts may come up in real documents.

For example, `babel` may want to provide hooks such as `babel/afterextras/⟨language⟩`. Language support in `babel` is often done through external language packages. Thus doing the activation for all languages inside the core `babel` code is not a viable approach. Instead it needs to be done by each language package (or by the user who wants to use a particular hook).

Because the hooks are not declared with `\NewHook` their names should be carefully chosen to ensure that they are (likely to be) unique. Best practice is to include the package or command name, as was done in the `babel` example above.

Generic hooks defined in this way are always normal hooks (i.e., you can’t implement reversed hooks this way). This is a deliberate limitation, because it speeds up the processing considerably.

2.7 Private L^AT_EX kernel hooks

There are a few places where it is absolutely essential for L^AT_EX to function correctly that code is executed in a precisely defined order. Even that could have been implemented with the hook management (by adding various rules to ensure the appropriate ordering with respect to other code added by packages). However, this makes every document unnecessary slow, because there has to be sorting even though the result is predetermined. Furthermore it forces package writers to unnecessarily add such rules if they add further code to the hook (or break L^AT_EX).

For that reason such code is not using the hook management, but instead private kernel commands directly before or after a public hook with the following naming convention:

`\@kernel@before@{hook}` or `\@kernel@after@{hook}`. For example, in `\enddocument` you find

```
\UseHook{enddocument}%
\@kernel@after@enddocument
```

which means first the user/package-accessible `enddocument` hook is executed and then the internal kernel hook. As their name indicates these kernel commands should not be altered by third-party packages, so please refrain from that in the interest of stability and instead use the public hook next to it.⁸

2.8 Legacy L^AT_EX 2_E interfaces

L^AT_EX 2_E offered a small number of hooks together with commands to add to them. They are listed here and are retained for backwards compatibility.

With the new hook management, several additional hooks have been added to L^AT_EX and more will follow. See the next section for what is already available.

`\AtBeginDocument` `\AtBeginDocument [{label}] {code}`

If used without the optional argument `{label}`, it works essentially like before, i.e., it is adding `{code}` to the hook `begindocument` (which is executed inside `\begin{document}`). However, all code added this way is labeled with the label `top-level` (see section 2.1.6) if done outside of a package or class or with the package/class name if called inside such a file (see section 2.1.5).

This way one can add further code to the hook using `\AddToHook` or `\AtBeginDocument` using a different label and explicitly order the code chunks as necessary, e.g., run some code before or after another package's code. When using the optional argument the call is equivalent to running `\AddToHook {begindocument} [{label}] {code}`.

`\AtBeginDocument` is a wrapper around the `begindocument` hook (see section 3.2), which is a one-time hook. As such, after the `begindocument` hook is executed at `\begin{document}` any attempt to add `{code}` to this hook with `\AtBeginDocument` or with `\AddToHook` will cause that `{code}` to execute immediately instead. See section 2.5 for more on one-time hooks.

For important packages with known order requirement we may over time add rules to the kernel (or to those packages) so that they work regardless of the loading-order in the document.

`\AtEndDocument` `\AtEndDocument [{label}] {code}`

Like `\AtBeginDocument` but for the `enddocument` hook.

The few hooks that existed previously in L^AT_EX 2_E used internally commands such as `\@begindocumenthook` and packages sometimes augmented them directly rather than working through `\AtBeginDocument`. For that reason there is currently support for this, that is, if the system detects that such an internal legacy hook command contains code it adds it to the new hook system under the label `legacy` so that it doesn't get lost.

However, over time the remaining cases of direct usage need updating because in one of the future release of L^AT_EX we will turn this legacy support off, as it does unnecessary slow down the processing.

⁸As with everything in TeX there is no enforcement of this rule, and by looking at the code it is easy to find out how the kernel adds to them. The main reason of this section is therefore to say "please don't do that, this is unconfigurable code!"

3 L^AT_EX 2 _{ϵ} commands and environments augmented by hooks

In this section we describe the standard hooks that are now offered by L^AT_EX, or give pointers to other documents in which they are described. This section will grow over time (and perhaps eventually move to usrguide3).

3.1 Generic hooks

As stated earlier, with the exception of generic hooks, all hooks must be declared with `\NewHook` before they can be used. All generic hooks have names of the form “ $\langle type \rangle / \langle name \rangle / \langle position \rangle$ ”, where $\langle type \rangle$ is from the predefined list shown below, and $\langle name \rangle$ is the variable part whose meaning will depend on the $\langle type \rangle$. The last component, $\langle position \rangle$, has more complex possibilities: it can always be `before` or `after`; for `env` hooks, it can also be `begin` or `end`; and for `include` hooks it can also be `end`. Each specific hook is documented below, or in `ltcmdhooks-doc.pdf` or `ltfilehook-doc.pdf`.

The generic hooks provided by L^AT_EX belong to one of the six types:

env Hooks executed before and after environments – $\langle name \rangle$ is the name of the environment, and available values for $\langle position \rangle$ are `before`, `begin`, `end`, and `after`;

cmd Hooks added to and executed before and after commands – $\langle name \rangle$ is the name of the command, and available values for $\langle position \rangle$ are `before` and `after`;

file Hooks executed before and after reading a file – $\langle name \rangle$ is the name of the file (with extension), and available values for $\langle position \rangle$ are `before` and `after`;

package Hooks executed before and after loading packages – $\langle name \rangle$ is the name of the package, and available values for $\langle position \rangle$ are `before` and `after`;

class Hooks executed before and after loading classes – $\langle name \rangle$ is the name of the class, and available values for $\langle position \rangle$ are `before` and `after`;

include Hooks executed before and after `\included` files – $\langle name \rangle$ is the name of the included file (without the `.tex` extension), and available values for $\langle position \rangle$ are `before`, `end`, and `after`.

Each of the hooks above are detailed in the following sections and in linked documentation.

3.1.1 Generic hooks for all environments

Every environment $\langle env \rangle$ has now four associated hooks coming with it:

env/ $\langle env \rangle$ /before This hook is executed as part of `\begin{env}` as the very first action, in particular prior to starting the environment group. Its scope is therefore not restricted by the environment.

env/ $\langle env \rangle$ /begin This hook is executed as part of `\begin{env}` directly in front of the code specific to the environment start (e.g., the second argument of `\newenvironment`). Its scope is the environment body.

env/ $\langle env \rangle$ /end This hook is executed as part of `\end{env}` directly in front of the code specific to the end of the environment (e.g., the third argument of `\newenvironment`).

env/<env>/after This hook is executed as part of `\end` after the code specific to the environment end and after the environment group has ended. Its scope is therefore not restricted by the environment.

The hook is implemented as a reversed hook so if two packages add code to `env/<env>/before` and to `env/<env>/after` they can add surrounding environments and the order of closing them happens in the right sequence.

Generic environment hooks are never one-time hooks even with environments that are supposed to appear only once in a document.⁹ In contrast to other hooks there is also no need to declare them using `\NewHook`.

The hooks are only executed if `\begin{<env>}` and `\end{<env>}` is used. If the environment code is executed via low-level calls to `\<env>` and `\end{<env>}` (e.g., to avoid the environment grouping) they are not available. If you want them available in code using this method, you would need to add them yourself, i.e., write something like

```
\UseHook{env/quote/before}\quote  
...  
\endquote\UseHook{env/quote/after}
```

to add the outer hooks, etc.

Largely for compatibility with existing packages, the following four commands are also available to set the environment hooks; but for new packages we recommend directly using the hook names and `\AddToHook`.

\BeforeBeginEnvironment `\BeforeBeginEnvironment [<label>] {<env>} {<code>}`

This declaration adds to the `env/<env>/before` hook using the `<label>`. If `<label>` is not given, the `<default label>` is used (see section 2.1.5).

\AtBeginEnvironment `\AtBeginEnvironment [<label>] {<env>} {<code>}`

This is like `\BeforeBeginEnvironment` but it adds to the `env/<env>/begin` hook.

\AtEndEnvironment `\AtEndEnvironment [<label>] {<env>} {<code>}`

This is like `\BeforeBeginEnvironment` but it adds to the `env/<env>/end` hook.

\AfterEndEnvironment `\AfterEndEnvironment [<label>] {<env>} {<code>}`

This is like `\BeforeBeginEnvironment` but it adds to the `env/<env>/after` hook.

3.1.2 Generic hooks for commands

Similar to environments there are now (at least in theory) two generic hooks available for any L^AT_EX command. These are

cmd/<name>/before This hook is executed at the very start of the command execution.

cmd/<name>/after This hook is executed at the very end of the command body. It is implemented as a reversed hook.

In practice there are restrictions and especially the `after` hook works only with a subset of commands. Details about these restrictions are documented in `ltcmdhooks-doc.pdf` or with code in `ltcmdhooks-code.pdf`.

⁹Thus if one adds code to such hooks after the environment has been processed, it will only be executed if the environment appears again and if that doesn't happen the code will never get executed.

3.1.3 Generic hooks provided by file loading operations

There are several hooks added to L^AT_EX's process of loading file via its high-level interfaces such as `\input`, `\include`, `\usepackage`, `\RequirePackage`, etc. These are documented in `ltfilehook-doc.pdf` or with code in `ltfilehook-code.pdf`.

3.2 Hooks provided by `\begin{document}`

Until 2020 `\begin{document}` offered exactly one hook that one could add to using `\AtBeginDocument`. Experiences over the years have shown that this single hook in one place was not enough and as part of adding the general hook management system a number of additional hooks have been added at this point. The places for these hooks have been chosen to provide the same support as offered by external packages, such as `etoolbox` and others that augmented `\document` to gain better control.

Supported are now the following hooks (all of them one-time hooks):

`begindocument/before` This hook is executed at the very start of `\document`, one can think of it as a hook for code at the end of the preamble section and this is how it is used by `etoolbox`'s `\AtEndPreamble`.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

`begindocument` This hook is added to when using `\AtBeginDocument` and it is executed after the `.aux` file as be read in and most initialization are done, so they can be altered and inspected by the hook code. It is followed by a small number of further initializations that shouldn't be altered and are therefore coming later.

The hook should not be used to add material for typesetting as we are still in L^AT_EX's initialization phase and not in the document body. If such material needs to be added to the document body use the next hook instead.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

`begindocument/end` This hook is executed at the end of the `\document` code in other words at the beginning of the document body. The only command that follows it is `\ignorespaces`.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

The generic hooks executed by `\begin` also exist, i.e., `env/document/before` and `env/document/begin`, but with this special environment it is better use the dedicated one-time hooks above.

3.3 Hooks provided by `\end{document}`

L^AT_EX 2_& always provided `\AtEndDocument` to add code to the execution of `\end{document}` just in front of the code that is normally executed there. While this was a big improvement over the situation in L^AT_EX 2.09 it was not flexible enough for a number of use cases and so packages, such as `etoolbox`, `atveryend` and others patched `\enddocument` to add additional points where code could be hooked into.

Patching using packages is always problematical as leads to conflicts (code availability, ordering of patches, incompatible patches, etc.). For this reason a number of

additional hooks have been added to the `\enddocument` code to allow packages to add code in various places in a controlled way without the need for overwriting or patching the core code.

Supported are now the following hooks (all of them one-time hooks):

`enddocument` The hook associated with `\AtEndDocument`. It is immediately called at the beginning of `\enddocument`.

When this hook is executed there may be still unprocessed material (e.g., floats on the deferlist) and the hook may add further material to be typeset. After it, `\clearpage` is called to ensure that all such material gets typeset. If there is nothing waiting the `\clearpage` has no effect.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

`enddocument/afterlastpage` As the name indicates this hook should not receive code that generates material for further pages. It is the right place to do some final housekeeping and possibly write out some information to the `.aux` file (which is still open at this point to receive data, but since there will be no more pages you need to write to it using `\immediate\write`). It is also the correct place to set up any testing code to be run when the `.aux` file is re-read in the next step.

After this hook has been executed the `.aux` file is closed for writing and then read back in to do some tests (e.g., looking for missing references or duplicated labels, etc.).

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

`enddocument/afteraux` At this point, the `.aux` file has been reprocessed and so this is a possible place for final checks and display of information to the user. However, for the latter you might prefer the next hook, so that your information is displayed after the (possibly longish) list of files if that got requested via `\listfiles`.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

`enddocument/info` This hook is meant to receive code that write final information messages to the terminal. It follows immediately after the previous hook (so both could have been combined, but then packages adding further code would always need to also supply an explicit rule to specify where it should go).

This hook already contains some code added by the kernel (under the labels `kernel/filelist` and `kernel/warnings`), namely the list of files when `\listfiles` has been used and the warnings for duplicate labels, missing references, font substitutions etc.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

`enddocument/end` Finally, this hook is executed just in front of the final call to `\@@end`.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5). Is it even possible to add code after this one?

There is also the hook `\shipout/lastpage`. This hook is executed as part of the last `\shipout` in the document to allow package to add final `\special`'s to that page. Where this hook is executed in relation to those from the above list can vary from document to document. Furthermore to determine correctly which of the `\shipouts` is the last one, L^AT_EX needs to be run several times, so initially it might get executed on the wrong page. See section 3.4 for where to find the details.

It is in also possible to use the generic `env/document/end` hook which is executed by `\end`, i.e., just in front of the first hook above. Note however that the other generic `\end` environment hook, i.e., `env/document/after` will never get executed, because by that time L^AT_EX has finished the document processing.

3.4 Hooks provided by `\shipout` operations

There are several hooks and mechanisms added to L^AT_EX's process of generating pages. These are documented in `ltshipout-doc.pdf` or with code in `ltshipout-code.pdf`.

3.5 Hooks provided for paragraphs

The paragraph processing has been augmented to include a number of internal and public hooks. These are documented in `ltpara-doc.pdf` or with code in `ltpara-code.pdf`.

3.6 Hooks provided in NFSS commands

In languages that need to support for more than one script in parallel (and thus several sets of fonts, e.g., supporting both Latin and Japanese fonts), NFSS font commands such as `\sffamily` need to switch both the Latin family to “Sans Serif” and in addition alter a second set of fonts.

To support this, several NFSS commands have hooks to which such support can be added.

rmfamily After `\rmfamily` has done its initial checks and prepared a font series update, this hook is executed before `\selectfont`.

sffamily This is like the `rmfamily` hook, but for the `\sffamily` command.

ttfamily This is like the `rmfamily` hook, but for the `\ttfamily` command.

normalfont The `\normalfont` command resets the font encoding, family, series and shape to their document defaults. It then executes this hook and finally calls `\selectfont`.

expand@font@defaults The internal `\expand@font@defaults` command expands and saves the current defaults for the meta families (rm/sf/tt) and the meta series (bf/md). If the NFSS machinery has been augmented, e.g., for Chinese or Japanese fonts, then further defaults may need to be set at this point. This can be done in this hook which is executed at the end of this macro.

bfseries/defaults, bfseries If the `\bfdefault` was explicitly changed by the user, its new value is used to set the bf series defaults for the meta families (rm/sf/tt) when `\bfseries` is called. The `bfseries/defaults` hook allows further adjustments to be made in this case. This hook is only executed if such a change is detected. In contrast, the `bfseries` hook is always executed just before `\selectfont` is called to change to the new series.

mdseries/defaults, **mdseries** These two hooks are like the previous ones but they are in the `\mdseries` command.

selectfont This hook is executed inside `\selectfont`, after the current values for *encoding*, *family*, *series*, *shape*, and *size* are evaluated and the new font is selected (and if necessary loaded). After the hook has executed, NFSS will still do any updates necessary for a new *size* (such as changing the size of `\strut`) and any updates necessary to a change in *encoding*.

This hook is intended for use cases where, in parallel to a change in the main font, some other fonts need to be altered (e.g., in CJK processing where you may need to deal with several different alphabets).

4 The Implementation

```
1 (@@=hook)
2 (*2ekernel | latexrelease)
3 \ExplSyntaxOn
4 (latexrelease) \NewModuleRelease{2020/10/01}{lthooks}
5 (latexrelease)           {The~hook~management~system}
```

4.1 Debugging

`\g__hook_debug_bool` Holds the current debugging state.

```
6 \bool_new:N \g__hook_debug_bool
```

(*End definition for* `\g__hook_debug_bool`.)

```
\hook_debug_on: Turns debugging on and off by redefining \__hook_debug:n.
\hook_debug_off: 
\__hook_debug:n
\__hook_debug_gset:
```

7 \cs_new_eq:NN __hook_debug:n \use_none:n
8 \cs_new_protected:Npn \hook_debug_on:
9 {
10 \bool_gset_true:N \g__hook_debug_bool
11 __hook_debug_gset:
12 }
13 \cs_new_protected:Npn \hook_debug_off:
14 {
15 \bool_gset_false:N \g__hook_debug_bool
16 __hook_debug_gset:
17 }
18 \cs_new_protected:Npn __hook_debug_gset:
19 {
20 \cs_gset_protected:Npx __hook_debug:n ##1
21 { \bool_if:NT \g__hook_debug_bool {##1} }
22 }

(*End definition for* `\hook_debug_on:` and others. These functions are documented on page 193.)

4.2 Borrowing from internals of other kernel modules

`__hook_str_compare:nn` Private copy of `__str_if_eq:nn`

```
23 \cs_new_eq:NN \__hook_str_compare:nn \__str_if_eq:nn
```

(*End definition for* `__hook_str_compare:nn`.)

4.3 Declarations

<code>\l_hook_tmpa_bool</code>	Scratch boolean used throughout the package.
	<small>24 \bool_new:N \l_hook_tmpa_bool</small>
	<i>(End definition for \l_hook_tmpa_bool.)</i>
<code>\l_hook_return_tl</code>	Scratch variables used throughout the package.
	<small>25 \tl_new:N \l_hook_return_tl</small>
	<small>26 \tl_new:N \l_hook_tmpa_tl</small>
	<small>27 \tl_new:N \l_hook_tmpb_tl</small>
	<i>(End definition for \l_hook_return_tl, \l_hook_tmpa_tl, and \l_hook_tmpb_tl.)</i>
<code>\g_hook_all_seq</code>	In a few places we need a list of all hook names ever defined so we keep track if them in this sequence.
	<small>28 \seq_new:N \g_hook_all_seq</small>
	<i>(End definition for \g_hook_all_seq.)</i>
<code>\l_hook_cur_hook_tl</code>	Stores the name of the hook currently being sorted.
	<small>29 \tl_new:N \l_hook_cur_hook_tl</small>
	<i>(End definition for \l_hook_cur_hook_tl.)</i>
<code>\l_hook_work_prop</code>	A property list holding a copy of the <code>\g_hook_<hook>_code_prop</code> of the hook being sorted to work on, so that changes don't act destructively on the hook data structure.
	<small>30 \prop_new:N \l_hook_work_prop</small>
	<i>(End definition for \l_hook_work_prop.)</i>
<code>\g_hook_used_prop</code>	All hooks that receive code (for use in debugging display).
	<small>31 \prop_new:N \g_hook_used_prop</small>
	<i>(End definition for \g_hook_used_prop.)</i>
<code>\g_hook_hook_curr_name_tl</code>	Default label used for hook commands, and a stack to keep track of packages within packages.
<code>\g_hook_name_stack_seq</code>	
	<small>32 \tl_new:N \g_hook_hook_curr_name_tl</small>
	<small>33 \seq_new:N \g_hook_name_stack_seq</small>
	<i>(End definition for \g_hook_hook_curr_name_tl and \g_hook_name_stack_seq.)</i>
<code>_hook_tmp:w</code>	Temporary macro for generic usage.
	<small>34 \cs_new_eq:NN _hook_tmp:w ?</small>
	<i>(End definition for _hook_tmp:w.)</i>
<code>\tl_gremove_once:Nx</code>	Some variants of <code>expl3</code> functions.
<code>\tl_show:x</code>	<i>FMi: should probably be moved to expl3</i>
<code>\tl_log:x</code>	
	<small>35 \cs_generate_variant:Nn \tl_gremove_once:Nn { Nx }</small>
	<small>36 \cs_generate_variant:Nn \tl_show:n { x }</small>
	<small>37 \cs_generate_variant:Nn \tl_log:n { x }</small>
	<i>(End definition for \tl_gremove_once:Nx, \tl_show:x, and \tl_log:x.)</i>

\s__hook_mark Scan mark used for delimited arguments.
 38 \scan_new:N \s__hook_mark
(End definition for \s__hook_mark.)

__hook_clean_to_scan:w Removes tokens until the next \s__hook_mark.
 39 \cs_new:Npn __hook_clean_to_scan:w #1 \s__hook_mark { }
(End definition for __hook_clean_to_scan:w.)

__hook_tl_set:Nn __hook_tl_set:Nx __hook_tl_set:cn __hook_tl_set:cx Private copies of a few expl3 functions. l3debug will only add debugging to the public names, not to these copies, so we don't have to use \debug_suspend: and \debug_resume: everywhere.
 Functions like __hook_tl_set:Nn have to be redefined, rather than copied because in expl3 they use __kernel_tl_(g)set:Nx, which is also patched by l3debug.
 40 \cs_new_protected:Npn __hook_tl_set:Nn #1#2
 41 { \cs_set_nopar:Npx #1 { __kernel_exp_not:w {#2} } }
 42 \cs_new_protected:Npn __hook_tl_set:Nx #1#2
 43 { \cs_set_nopar:Npx #1 {#2} }
 44 \cs_generate_variant:Nn __hook_tl_set:Nn { c }
 45 \cs_generate_variant:Nn __hook_tl_set:Nx { c }
(End definition for __hook_tl_set:Nn.)

__hook_tl_gset:Nn __hook_tl_gset:No __hook_tl_gset:Nx __hook_tl_gset:cn __hook_tl_gset:co __hook_tl_gset:cx Same as above.
 46 \cs_new_protected:Npn __hook_tl_gset:Nn #1#2
 47 { \cs_gset_nopar:Npx #1 { __kernel_exp_not:w {#2} } }
 48 \cs_new_protected:Npn __hook_tl_gset:No #1#2
 49 { \cs_gset_nopar:Npx #1 { __kernel_exp_not:w \exp_after:wN {#2} } }
 50 \cs_new_protected:Npn __hook_tl_gset:Nx #1#2
 51 { \cs_gset_nopar:Npx #1 {#2} }
 52 \cs_generate_variant:Nn __hook_tl_gset:Nn { c }
 53 \cs_generate_variant:Nn __hook_tl_gset:No { c }
 54 \cs_generate_variant:Nn __hook_tl_gset:Nx { c }
(End definition for __hook_tl_gset:Nn.)

__hook_tl_gput_right:Nn __hook_tl_gput_right:No __hook_tl_gput_right:cn Same as above.
 55 \cs_new_protected:Npn __hook_tl_gput_right:Nn #1#2
 56 { __hook_tl_gset:Nx #1 { __kernel_exp_not:w \exp_after:wN { #1 #2 } } }
 57 \cs_generate_variant:Nn __hook_tl_gput_right:Nn { No, cn }
(End definition for __hook_tl_gput_right:Nn.)

__hook_tl_gput_left:Nn __hook_tl_gput_left:No Same as above.
 58 \cs_new_protected:Npn __hook_tl_gput_left:Nn #1#2
 59 {
 60 { __hook_tl_gset:Nx #1
 61 { __kernel_exp_not:w {#2} __kernel_exp_not:w \exp_after:wN {#1} } }
 62 }
 63 \cs_generate_variant:Nn __hook_tl_gput_left:Nn { No }
(End definition for __hook_tl_gput_left:Nn.)

```

\__hook_tl_gset_eq:NN  Same as above.
64 \cs_new_eq:NN \__hook_tl_gset_eq:NN \tl_gset_eq:NN
(End definition for \__hook_tl_gset_eq:NN.)

\__hook_tl_gclear:N  Same as above.
\__hook_tl_gclear:c 65 \cs_new_protected:Npn \__hook_tl_gclear:N #1
66 { \__hook_tl_gset_eq:NN #1 \c_empty_tl }
67 \cs_generate_variant:Nn \__hook_tl_gclear:N { c }

(End definition for \__hook_tl_gclear:N.)

```

4.4 Providing new hooks

4.4.1 The data structures of a hook

\g_@@_{hook}_code_prop Hooks have a name (called *<hook>* in the description below) and for each hook we have \g_@@_{hook} to provide a number of data structures. These are

\g_@@_next_{hook} \g__hook_{hook}_code_prop A property list holding the code for the hook in separate chunks. The keys are by default the package names that add code to the hook, but it is possible for packages to define other keys.

\g__hook_{hook}_rule_{label1}|{label2}_tl A token list holding the relation between *<label1>* and *<label2>* in the *<hook>*. The *<labels>* are lexically (reverse) sorted to ensure that two labels always point to the same token list. For global rules, the *<hook>* name is ??.

__hook_{hook} The code that is actually executed when the hook is called in the document is stored in this token list. It is constructed from the code chunks applying the information. This token list is named like that so that in case of an error inside the hook, the reported token list in the error is shorter, and to make it simpler to normalize hook names in __hook_make_name:n.

\g__hook_{hook}_reversed_t1 Some hooks are “reversed”. This token list stores a - for such hook so that it can be identified. The - character is used because *<reversed>*1 is +1 for normal hooks and -1 for reversed ones.

\g__hook_{hook}_declared_t1 This token list serves as marker for the hook being officially declared. Its existence is tested to raise an error in case another declaration is attempted.

__hook_toplevel_{hook} This token list stores the code inserted in the hook from the user’s document, in the top-level label. This label is special, and doesn’t participate in sorting. Instead, all code is appended to it and executed after (or before, if the hook is reversed) the normal hook code, but before the next code chunk.

__hook_next_{hook} Finally there is extra code (normally empty) that is used on the next invocation of the hook (and then deleted). This can be used to define some special behavior for a single occasion from within the document. This token list follows the same naming scheme than the main __hook_{hook} token list. It is called __hook_next_{hook} rather than __hook_next_{hook} because otherwise a hook whose name is next_{hook} would clash with the next code-token list of the hook called *<hook>*.

4.4.2 On the existence of hooks

A hook may be in different states of existence. Here we give an overview of the internal commands to set up hooks and explain how the different states are distinguished. The actual implementation then follows in subsequent sections.

One problem we have to solve is that we need to be able to add code to hooks (e.g., with `\AddToHook`) even if that code has not yet been declared. For example, one package needs to write into a hook of another package, but that package may not get loaded, or is loaded only later. Another problem is that most hooks, but not the generic hooks, require a declaration.

We therefore distinguish the following states for a hook, which are managed by four different tests: structure existence (`__hook_if_structure_exist:nTF`), creation (`__hook_if_usable:nTF`), declaration (`__hook_if_declared:nTF`) and disabled or not (`__hook_if_disabled:nTF`)

not existing Nothing is known about the hook so far. This state can be detected with `__hook_if_structure_exist:nTF` (which uses the false branch).

In this state the hook can be declared, disabled, rules can be defined or code could be added to it, but it is not possible to use the hook (with `\UseHook`).

basic data structure set up A hook is in this state when its basic data structure has been set up (using `__hook_init_structure:n`). The data structure setup happens automatically when commands such as `\AddToHook` are used and the hook is at that point in state “not existing”.

In this state the four tests give the following results:

```
\_\_hook\_if\_structure\_exist:nTF returns true.  
    \_\_hook\_if\_usable:nTF returns false.  
    \_\_hook\_if\_declared:nTF returns false.  
    \_\_hook\_if\_disabled:nTF returns false.
```

The allowed actions are the same as in the “not existing” state.

declared A hook is in this state if it is not disabled and was explicitly declared (e.g., with `\NewHook`). In this case the four tests give the following results:

```
\_\_hook\_if\_structure\_exist:nTF returns true.  
    \_\_hook\_if\_usable:nTF returns true.  
    \_\_hook\_if\_declared:nTF returns true.  
    \_\_hook\_if\_disabled:nTF returns false.
```

usable A hook is in this state if it is not disabled, was not explicitly declared but nevertheless is allowed to be used (with `\UseHook` or `\hook_use:n`). This state is only possible for generic hooks as they do not need to be declared. Therefore such hooks move directly from state “not existing” to “usable” the moment a declaration such as `\AddToHook` wants to add to the hook data structure. In this state the tests give the following results:

```
\_\_hook\_if\_structure\_exist:nTF returns true.  
    \_\_hook\_if\_usable:nTF returns true.
```

```

\__hook_if_declared:nTF returns false.
\__hook_if_disabled:nTF returns false.

```

disabled A generic hook in any state is moved to this state when `\DisableGenericHook` is used. This changes the tests to give the following results:

```

\__hook_if_structure_exist:nTF unchanged.
    \__hook_if_usable:nTF returns false.
    \__hook_if_declared:nTF returns true.
    \__hook_if_disabled:nTF returns true.

```

The structure test is unchanged (if the hook was unknown before it is `false`, otherwise `true`). The usable test returns `false` so that any `\UseHook` will bypass the hook from now on. The declared test returns `true` so that any further `\NewHook` generates an error and the disabled test returns `true` so that `\AddToHook` can return an error.

FMi: maybe it should do this only after begin document?

4.4.3 Setting hooks up

\hook_new:n
`_hook_new:n` The `\hook_new:n` declaration declares a new hook and expects the hook `<name>` as its argument, e.g., `\begindocument`.

```

68 \cs_new_protected:Npn \hook_new:n #1
69   { \__hook_normalize_hook_args:Nn \_hook_new:n {#1} }
70 \cs_new_protected:Npn \_hook_new:n #1
71   {

```

We check if the hook was already *explicitly* declared with `\hook_new:n`, and if it already exists we complain, otherwise set the “created” flag for the hook so that it errors next time `\hook_new:n` is used.

```

72   \__hook_if_declared:nTF {#1}
73     { \msg_error:nnn { hooks } { exists } {#1} }
74   {
75     \tl_new:c { g__hook_#1_declared_tl }
76     \__hook_make_usable:n {#1}
77   }
78 }

```

(End definition for `\hook_new:n` and `_hook_new:n`. This function is documented on page 191.)

`_hook_make_usable:n` This initializes all hook data structures for the hook but if used on its own doesn’t mark the hook as declared (as `\hook_new:n` does, so a later `\hook_new:n` on that hook will not result in an error). This command is internally used by `\hook_gput_code:n` when adding code to a generic hook.

```

79 \cs_new_protected:Npn \_hook_make_usable:n #1
80   {

```

Now we check if the hook’s data structure can be safely created without `expl3` raising errors, then we add the hook name to the list of all hooks and allocate the necessary data structures for the new hook, otherwise just do nothing.

```

81   \tl_if_exist:cF { __hook~#1 }
82   {
83     \seq_gput_right:Nn \g__hook_all_seq {#1}

```

This is only used by the actual code of the current hook, so declare it normally:

```
84     \tl_new:c { __hook~#1 }
```

Now ensure that the base data structure for the hook exists:

```
85     \__hook_init_structure:n {#1}
```

The `\g__hook_<hook>.labels_clist` holds the sorted list of labels (once it got sorted). This is used only for debugging.

```
86     \clist_new:c { g__hook_#1_labels_clist }
```

Some hooks should reverse the default order of code chunks. To signal this we have a token list which is empty for normal hooks and contains a `-` for reversed hooks.

```
87     \tl_new:c { g__hook_#1_reversed_tl }
```

The above is all in L3 convention, but we also provide an interface to legacy L^AT_EX 2 ε hooks of the form `\@...hook`, e.g., `\begin{document}hook`. There have been a few of them and they have been added to using `\g@addto@macro`. If there exists such a macro matching the name of the new hook, i.e., `\@<hook-name>hook` and it is not empty then we add its contents as a code chunk under the label `legacy`.

Warning: this support will vanish in future releases!

```
88     \__hook_include_legacy_code_chunk:n {#1}
89     }
90 }
```

(End definition for `__hook_make_usable:n`.)

`__hook_init_structure:n` This function declares the basic data structures for a hook without explicit declaring the hook itself. This is needed to allow adding to undeclared hooks. Here it is unnecessary to check whether all variables exist, since all three are declared at the same time (either all of them exist, or none).

It creates the hook code pool (`\g__hook_<hook>.code_prop`) and the `top-level` and `next` token lists. A hook is initialized with `__hook_init_structure:n` the first time anything is added to it. Initializing a hook just with `__hook_init_structure:n` will not make it usable with `\hook_use:n`.

```
91 \cs_new_protected:Npn \__hook_init_structure:n #1
92 {
93     \__hook_if_structure_exist:nF {#1}
94     {
95         \prop_new:c { g__hook_#1_code_prop }
96         \tl_new:c { __hook_toplevel~#1 }
97         \tl_new:c { __hook_next~#1 }
98     }
99 }
```

(End definition for `__hook_init_structure:n`.)

`\hook_new_reversed:n` Declare a new hook. The default ordering of code chunks is reversed, signaled by setting the token list to a minus sign.

```
100 \cs_new_protected:Npn \hook_new_reversed:n #1
101   { \__hook_normalize_hook_args:Nn \__hook_new_reversed:n {#1} }
102 \cs_new_protected:Npn \__hook_new_reversed:n #1
103   {
104     \__hook_new:n {#1}
```

If the hook already exists the above will generate an error message, so the next line should be executed (but it is — too bad).

```
105     \tl_gset:cn { g__hook_#1_reversed_tl } { - }
106 }
```

(End definition for \hook_new_reversed:n and _hook_new_reversed:n. This function is documented on page 191.)

\hook_new_pair:nn A shorthand for declaring a normal and a (matching) reversed hook in one go.

```
107 \cs_new_protected:Npn \hook_new_pair:nn #1#2
108   { \hook_new:n {#1} \hook_new_reversed:n {#2} }
```

(End definition for \hook_new_pair:nn. This function is documented on page 191.)

_hook_include_legacy_code_chunk:n

The L^AT_EX legacy concept for hooks uses with hooks the following naming scheme in the code: \@...hook.

If this macro is not empty we add it under the label `legacy` to the current hook and then empty it globally. This way packages or classes directly manipulating commands such as \begindocumenthook still get their hook data added.

Warning: this support will vanish in future releases!

```
109 \cs_new_protected:Npn \_hook_include_legacy_code_chunk:n #1
110  {
```

If the macro doesn't exist (which is the usual case) then nothing needs to be done.

```
111 \tl_if_exist:cT { @#1hook }
```

Of course if the legacy hook exists but is empty, there is no need to add anything under `legacy` the legacy label.

```
112 {
113   \tl_if_empty:cF { @#1hook }
114   {
115     \exp_args:Nnnv \_hook_hook_gput_code_do:nnn {#1}
116       { legacy } { @#1hook }
```

Once added to the hook, we need to clear it otherwise it might get added again later if the hook data gets updated.

```
117   \_hook_tl_gclear:c { @#1hook }
118 }
119 }
120 }
```

(End definition for _hook_include_legacy_code_chunk:n.)

4.4.4 Disabling and providing hooks

\hook_disable_generic:n

_hook_disable:n
_hook_if_disabled:p:n

_hook_if_disabled:nTF

Disables a hook by creating its `\g__hook_{hook}_declared_tl` so that the hook errors when used with `\hook_new:n`, then it undefines `_hook_{hook}` so that it may not be executed.

This does not clear any code that may be already stored in the hook's structure, but doesn't allow adding more code. `_hook_if_disabled:nTF` uses that specific combination to check if the hook is disabled.

```
121 <latexrelease>\IncludeInRelease{2021/06/01}%
122 <latexrelease>      {\hook_disable_generic:n}{Disable~hooks}
```

```

123 \cs_new_protected:Npn \hook_disable_generic:n #1
124   { \__hook_normalize_hook_args:Nn \__hook_disable:n {#1} }
125 \cs_new_protected:Npn \__hook_disable:n #1
126   {
127     \tl_gclear_new:c { g__hook_#1_declared_tl }
128     \cs_undefine:c { __hook~#1 }
129   }
130 \prg_new_conditional:Npnn \__hook_if_disabled:n #1 { p, T, F, TF }
131   {
132     \bool_lazy_and:nnTF
133       { \tl_if_exist_p:c { g__hook_#1_declared_tl } }
134       { ! \tl_if_exist_p:c { __hook~#1 } }
135     { \prg_return_true: }
136     { \prg_return_false: }
137   }
138 \end{IncludeInRelease}
139 \IncludeInRelease{2020/10/01}
140 \hook_disable_generic:n}{Disable-hooks}
141 \end{IncludeInRelease}
142 \cs_new_protected:Npn \hook_disable_generic:n #1 {}
143 \end{IncludeInRelease}
144 \end{IncludeInRelease}

(End definition for \hook_disable_generic:n, \__hook_disable:n, and \__hook_if_disabled:nTF.
This function is documented on page 191.)

```

\hook_activate_generic:n The \hook_activate_generic:n declaration declares a new hook if it wasn't declared already, in which case it only checks that the already existing hook is not a reversed hook.

```

145 \IncludeInRelease{2021/06/01}%
146 \hook_activate_generic:n}{Providing~hooks}
147 \cs_new_protected:Npn \hook_activate_generic:n #1
148   { \__hook_normalize_hook_args:Nn \__hook_activate_generic:nn {#1} { } }
149 \cs_new_protected:Npn \__hook_activate_generic:nn #1 #2
150   {

```

If the hook to be activated was disabled we warn (for now — this may change).

```

151   \__hook_if_disabled:nTF {#1}
152     { \msg_warning:nnn { hooks } { activate-disabled } {#1} }

```

Otherwise we check if the hook is not declared, and if it isn't, figure out if it's reversed or not, then declare it accordingly.

```

153   {
154     \__hook_if_declared:nF {#1}
155     {
156       \tl_new:c { g__hook_#1_declared_tl }
157       \__hook_make_usable:n {#1}
158       \tl_gset:cx { g__hook_#1_reversed_tl }
159       { \__hook_if_generic_reversed:nT {#1} { - } }
160     }
161   }
162 }
```

(End definition for \hook_activate_generic:n and __hook_activate_generic:n. This function is documented on page 192.)

```

163 \end{IncludeInRelease}
```

```

164 〈latexrelease〉\IncludeInRelease{2020/10/01}
165 〈latexrelease〉                                {\hook_activate_generic:n}{Providing~hooks}
166 〈latexrelease〉
167 〈latexrelease〉\cs_new_protected:Npn \hook_activate_generic:n #1 {}
168 〈latexrelease〉
169 〈latexrelease〉\EndIncludeInRelease

```

4.5 Parsing a label

This macro checks if a label was given (not `\c_novalue_tl`), and if so, tries to parse the label looking for a leading `.` to replace by `_hook_currname_or_default:`.

```

170 \cs_new:Npn \_hook_parse_label_default:n #1
171 {
172     \tl_if_novalue:nTF {#1}
173     {
174         \_hook_currname_or_default:
175         \tl_trim_spaces_apply:nN {#1} \_hook_parse_dot_label:n
176     }

```

(*End definition for `_hook_parse_label_default:n`.*)

`_hook_parse_dot_label:n`
`_hook_parse_dot_label:w`
`_hook_parse_dot_label_cleanup:w`
`_hook_parse_dot_label_aux:w`

Start by checking if the label is empty, which raises an error, and uses the fallback value. If not, split the label at a `./`, if any, and check if no tokens are before the `./`, or if the only character is a `..`. If these requirements are fulfilled, the leading `.` is replaced with `_hook_currname_or_default:`. Otherwise the label is returned unchanged.

```

177 \cs_new:Npn \_hook_parse_dot_label:n #1
178 {
179     \tl_if_empty:nTF {#1}
180     {
181         \msg_expandable_error:nn { hooks } { empty-label }
182         \_hook_currname_or_default:
183     }
184     {
185         \str_if_eq:nnTF {#1} { . }
186         {
187             \_hook_currname_or_default:
188             \_hook_parse_dot_label:w #1 ./ \s__hook_mark
189         }
190     }
191     \cs_new:Npn \_hook_parse_dot_label:w #1 ./ #2 \s__hook_mark
192     {
193         \tl_if_empty:nTF {#1}
194         {
195             \_hook_parse_dot_label_aux:w #2 \s__hook_mark
196         }
197     }
198     \cs_new:Npn \_hook_parse_dot_label_cleanup:w #1 ./ \s__hook_mark {#1}
199     \cs_new:Npn \_hook_parse_dot_label_aux:w #1 ./ \s__hook_mark
200     {
201         \_hook_currname_or_default: / \_hook_make_name:n {#1}
202     }

```

(*End definition for `_hook_parse_dot_label:n` and others.*)

`__hook_currname_or_default:` This uses `\g__hook_hook_curr_name_tl` if it is set, otherwise it tries `\@currname`. If neither is set, it raises an error and uses the fallback value `label-missing`.

```

202 \cs_new:Npn \__hook_currname_or_default:
203 {
204     \tl_if_empty:NTF \g__hook_hook_curr_name_tl
205     {
206         \tl_if_empty:NTF \@currname
207         {
208             \msg_expandable_error:nnn { latex2e } { should-not-happen }
209             {
210                 \Empty~default-label.
211             }
212             \__hook_make_name:n { label-missing }
213         }
214         { \g__hook_hook_curr_name_tl }
215     }

```

(End definition for `__hook_currname_or_default`.)

`__hook_make_name:n` This provides a standard sanitization of a hook's name. It uses `\cs:w` to build a control sequence out of the hook name, then uses `\cs_to_str:N` to get the string representation of that, without the escape character. `\cs:w`-based expansion is used instead of `e`-based because Unicode characters don't behave well inside `\expanded`. The macro adds the `__hook_` prefix to the hook name to reuse the hook's code token list to build the csname and avoid leaving "public" control sequences defined (as `\relax`) in TeX's memory.

```

216 \cs_new:Npn \__hook_make_name:n #1
217 {
218     \exp_after:wN \exp_after:wN \exp_after:wN \__hook_make_name:w
219     \exp_after:wN \token_to_str:N \cs:w __hook~ #1 \cs_end:
220 }
221 \exp_last_unbraced:NNNNo
222 \cs_new:Npn \__hook_make_name:w #1 \tl_to_str:n { __hook~ } { }

```

(End definition for `__hook_make_name:n` and `__hook_make_name:w`.)

`__hook_normalize_hook_args:Nn`
`__hook_normalize_hook_args:Nnn`
`__hook_normalize_hook_rule_args:Nnnnn`
`__hook_normalize_hook_args_aux:Nn`

This is the standard route for normalizing hook and label arguments. The main macro does the entire operation within a group so that csnames made by `__hook_make_name:n` are wiped off before continuing. This means that this function cannot be used for `\hook_use:n!`

```

223 \cs_new_protected:Npn \__hook_normalize_hook_args_aux:Nn #1 #2
224 {
225     \group_begin:
226     \use:e
227     {
228         \group_end:
229         \exp_not:N #1 #2
230     }
231 }
232 \cs_new_protected:Npn \__hook_normalize_hook_args:Nn #1 #2
233 {
234     \__hook_normalize_hook_args_aux:Nn #1
235     { { \__hook_parse_label_default:n {#2} } }
236 }

```

```

237 \cs_new_protected:Npn \__hook_normalize_hook_args:Nnn #1 #2 #3
238   {
239     \__hook_normalize_hook_args_aux:Nn #1
240     {
241       { \__hook_parse_label_default:n {#2} }
242       { \__hook_parse_label_default:n {#3} }
243     }
244   }
245 \cs_new_protected:Npn \__hook_normalize_hook_rule_args:Nnnnn #1 #2 #3 #4 #5
246   {
247     \__hook_normalize_hook_args_aux:Nn #1
248     {
249       { \__hook_parse_label_default:n {#2} }
250       { \__hook_parse_label_default:n {#3} }
251       { \tl_trim_spaces:n {#4} }
252       { \__hook_parse_label_default:n {#5} }
253     }
254   }

```

(End definition for `__hook_normalize_hook_args:Nn` and others.)

`__hook_curr_name_push:n`
`__hook_curr_name_push_aux:n`
`__hook_curr_name_pop:`
`__hook_end_document_label_check:`

The token list `\g__hook_hook_curr_name_tl` stores the name of the current package/file to be used as the default label in hooks. Providing a consistent interface is tricky because packages can be loaded within packages, and some packages may not use `\SetDefaultHookLabel` to change the default label (in which case `\@currname` is used).

To pull that one off, we keep a stack that contains the default label for each level of input. The bottom of the stack contains the default label for the **top-level** (this stack should never go empty). If we're building the format, set the default label to be **top-level**:

```
255 \tl_gset:Nn \g__hook_hook_curr_name_tl { top-level }
```

Then, in case we're in `\textralatexrelease` we push something on the stack to support roll forward. But in some rare cases, `\textralatexrelease` may be loaded inside another package (notably `\platajaxrelease`), so we'll first push the **top-level** entry:

```
256 <\textralatexrelease> \seq_if_empty:NT \g__hook_name_stack_seq
257 <\textralatexrelease> { \seq_gput_right:Nn \g__hook_name_stack_seq { top-level } }
```

then we dissect the `\@currnamestack`, adding `\@currname` to the stack:

```

258 <\textralatexrelease> \cs_set_protected:Npn \__hook_tmp:w #1 #2 #3
259 <\textralatexrelease> {
260   <\textralatexrelease> \quark_if_recursion_tail_stop:n {#1}
261   <\textralatexrelease> \seq_gput_right:Nn \g__hook_name_stack_seq {#1}
262   <\textralatexrelease> \__hook_tmp:w
263 <\textralatexrelease> }
264 <\textralatexrelease> \exp_after:wN \__hook_tmp:w \@currnamestack
265 <\textralatexrelease> \q_recursion_tail \q_recursion_tail
266 <\textralatexrelease> \q_recursion_tail \q_recursion_stop

```

and finally set the default label to be the `\@currname`:

```
267 <\textralatexrelease> \tl_gset:Nx \g__hook_hook_curr_name_tl { \@currname }
268 <\textralatexrelease> \seq_gpop_right:NN \g__hook_name_stack_seq \l__hook_tmpa_t1
```

Two commands keep track of the stack: when a file is input, `__hook_curr_name_push:n` pushes the current default label onto the stack and sets the new default label (all in one go):

```

269 \cs_new_protected:Npn \__hook_curr_name_push:n #1
270   { \exp_args:Nx \__hook_curr_name_push_aux:n { \__hook_make_name:n {#1} } }
271 \cs_new_protected:Npn \__hook_curr_name_push_aux:n #1
272   {
273     \tl_if_blank:nTF {#1}
274       { \msg_error:nn { hooks } { no-default-label } }
275       {
276         \str_if_eq:nnTF {#1} { top-level }
277           {
278             \msg_error:nnnn { hooks } { set-top-level }
279             { to } { PushDefaultHookLabel } {#1}
280           }
281           {
282             \seq_gpush:NV \g__hook_name_stack_seq \g__hook_hook_curr_name_tl
283             \tl_gset:Nn \g__hook_hook_curr_name_tl {#1}
284           }
285         }
286       }

```

and when an input is over, the topmost item of the stack is popped, since that label will not be used again, and `\g__hook_hook_curr_name_tl` is updated to equal the now topmost item of the stack:

```

287 \cs_new_protected:Npn \__hook_curr_name_pop:
288   {
289     \seq_gpop:NNTF \g__hook_name_stack_seq \l__hook_return_tl
290       { \tl_gset_eq:NN \g__hook_hook_curr_name_tl \l__hook_return_tl }
291       { \msg_error:nn { hooks } { extra-pop-label } }
292   }

```

At the end of the document we want to check if there was no `__hook_curr_name_push:n` without a matching `__hook_curr_name_pop:` (not a critical error, but it might indicate that something else is not quite right):

```

293 \tl_gput_right:Nn \@kernel@after@enddocument@afterlastpage
294   { \__hook_end_document_label_check: }
295 \cs_new_protected:Npn \__hook_end_document_label_check:
296   {
297     \seq_gpop:NNT \g__hook_name_stack_seq \l__hook_return_tl
298     {
299       \msg_error:nnx { hooks } { missing-pop-label }
300       { \g__hook_hook_curr_name_tl }
301       \tl_gset_eq:NN \g__hook_hook_curr_name_tl \l__hook_return_tl
302       \__hook_end_document_label_check:
303     }
304   }

```

The token list `\g__hook_hook_curr_name_tl` is but a mirror of the top of the stack.

Now define a wrapper that replaces the top of the stack with the argument, and updates `\g__hook_hook_curr_name_tl` accordingly.

```

305 \cs_new_protected:Npn \__hook_set_default_hook_label:n #1
306   {
307     \seq_if_empty:NTF \g__hook_name_stack_seq
308       {
309         \msg_error:nnnn { hooks } { set-top-level }
310         { for } { SetDefaultHookLabel } {#1}

```

```

311      }
312      { \exp_args:Nx \__hook_set_default_label:n { \__hook_make_name:n {#1} } }
313    }
314 \cs_new_protected:Npn \__hook_set_default_label:n #1
315  {
316    \str_if_eq:nnTF {#1} { top-level }
317    {
318      \msg_error:nnnn { hooks } { set-top-level }
319      { to } { SetDefaultHookLabel } {#1}
320    }
321    { \tl_gset:Nn \g__hook_hook_curr_name_tl {#1} }
322  }

```

(End definition for `__hook_curr_name_push:n` and others.)

4.6 Adding or removing hook code

`\hook_gput_code:nnn`
`__hook_gput_code:nnn`
`__hook_hook_gput_code_do:nnn`

With `\hook_gput_code:nnn{<hook>}{<label>}{<code>}` a chunk of `<code>` is added to an existing `<hook>` labeled with `<label>`.

```

323 \cs_new_protected:Npn \hook_gput_code:nnn #1 #2
324   { \__hook_normalize_hook_args:Nnn \__hook_gput_code:nnn {#1} {#2} }
325 \cs_new_protected:Npn \__hook_gput_code:nnn #1 #2 #3
326  {

```

First check if the code should be executed immediately, rather than stored:

```

327   \__hook_if_execute_immediately:nTF {#1}
328     {#3}
329   {

```

Then check if the hook is usable.

```

330     \__hook_if_usable:nTF {#1}

```

If so we simply add (or append) the new code to the property list holding different chunks for the hook. At `\begin{document}` this is then sorted into a token list for fast execution.

```

331   {
332     \__hook_hook_gput_code_do:nnn {#1} {#2} {#3}

```

However, if there is an update within the document we need to alter this execution code which is done by `__hook_update_hook_code:n`. In the preamble this does nothing.

```

333   \__hook_update_hook_code:n {#1}
334 }

```

If the hook is not usable, before giving up, check if it's not disabled and otherwise try to declare it as a generic hook, if its name matches one of the valid patterns.

```

335   {
336     \__hook_if_disabled:nTF {#1}
337     { \msg_error:nn { hooks } { hook-disabled } {#1} }
338     { \__hook_try_declaring_generic_hook:nnn {#1} {#2} {#3} }
339   }
340 }
341 }

```

This macro will unconditionally add a chunk of code to the given hook.

```

342 \cs_new_protected:Npn \__hook_hook_gput_code_do:nnn #1 #2 #3
343  {

```

However, first some debugging info if debugging is enabled:

```

344     \__hook_debug:n{\iow_term:x{****~ Add~ to~
345             \__hook_if_usable:nF {#1} { undeclared~ }
346             hook~ #1~ (#2)
347             \on@line\space <-- \tl_to_str:n{#3}} }

```

Then try to get the code chunk labeled #2 from the hook. If there's code already there, then append #3 to that, otherwise just put #3. If the current label is **top-level**, the code is added to a dedicated token list `__hook_toplevel<hook>` that goes at the end of the hook (or at the beginning, for a reversed hook), just before `__hook_next<hook>`.

```

348     \str_if_eq:nnTF {#2} { top-level }
349     {
350         \str_if_eq:eeTF { top-level } { \__hook_currname_or_default: }
351         {

```

If the hook's basic structure does not exist, we need to declare it with `__hook_init_-structure:n`.

```

352         \__hook_init_structure:n {#1}
353         \__hook_tl_gput_right:cn { __hook_toplevel~#1 } {#3}
354     }
355     { \msg_error:nnn { hooks } { misused-top-level } {#1} }
356 }
357 {
358     \prop_get:cnNTF { g__hook_#1_code_prop } {#2} \l__hook_return_tl
359     {
360         \prop_gput:cno { g__hook_#1_code_prop } {#2}
361         { \l__hook_return_tl #3 }
362     }
363     { \prop_gput:cnn { g__hook_#1_code_prop } {#2} {#3} }
364 }
365 }

```

(*End definition for \hook_gput_code:nnn, __hook_gput_code:nnn, and __hook_hook_gput_code_do:nnn. This function is documented on page 192.*)

`__hook_gput_undeclared_hook:nnn`

Often it may happen that a package *A* defines a hook `foo`, but package *B*, that adds code to that hook, is loaded before *A*. In such case we need to add code to the hook before its declared.

```

366 \cs_new_protected:Npn \__hook_gput_undeclared_hook:nnn #1 #2 #3
367   {
368     \__hook_init_structure:n {#1}
369     \__hook_hook_gput_code_do:nnn {#1} {#2} {#3}
370   }

```

(*End definition for __hook_gput_undeclared_hook:nnn.*)

`__hook_try_declaring_generic_hook:nnn`
`__hook_try_declaring_generic_next_hook:nn`

These entry-level macros just pass the arguments along to the common `__hook_try_-declaring_generic_hook:nNNnn` with the right functions to execute when some action is to be taken.

The wrapper `__hook_try_declaring_generic_hook:nnn` then defers `\hook_gput_code:nnn` if the generic hook was declared, or to `__hook_gput_undeclared_hook:nnn` otherwise (the hook was tested for existence before, so at this point if it isn't generic, it doesn't exist).

The wrapper `__hook_try_declaring_generic_next_hook:nn` for next-execution hooks does the same: it defers the code to `\hook_gput_next_code:nn` if the generic hook was declared, or to `__hook_gput_next_do:nn` otherwise.

```

371 <latexrelease>\IncludeInRelease{2021/11/15}{\_\_hook_try_declaring_generic_hook:nnn}
372 <latexrelease>                                {Standardise-generic-hook-names}
373 \cs_new_protected:Npn \_\_hook_try_declaring_generic_hook:nnn #1
374 {
375     \_\_hook_try_declaring_generic_hook:wnTF #1 // \scan_stop: {#1}
376     \hook_gput_code:nnn
377     \_\_hook_gput_undeclared_hook:nnn
378     {#1}
379 }
380 \cs_new_protected:Npn \_\_hook_try_declaring_generic_next_hook:nn #1
381 {
382     \_\_hook_try_declaring_generic_hook:wnTF #1 // \scan_stop: {#1}
383     \hook_gput_next_code:nn
384     \_\_hook_gput_next_do:nn
385     {#1}
386 }
387 <latexrelease>\EndIncludeInRelease
388 <latexrelease>\IncludeInRelease{2020/10/01}{\_\_hook_try_declaring_generic_hook:nnn}
389 <latexrelease>                                {Standardise-generic-hook-names}
390 \cs_new_protected:Npn \_\_hook_try_declaring_generic_hook:nnn #1
391 <latexrelease> {
392     \_\_hook_try_declaring_generic_hook:nNNnn {#1}
393     \hook_gput_code:nnn \_\_hook_gput_undeclared_hook:nnn
394 }
395 \cs_new_protected:Npn \_\_hook_try_declaring_generic_next_hook:nn #1
396 <latexrelease> {
397     \_\_hook_try_declaring_generic_hook:nNNnn {#1}
398     \hook_gput_next_code:nn \_\_hook_gput_next_do:nn
399 }

(End definition for \_\_hook_try_declaring_generic_hook:nnn and
 \_\_hook_try_declaring_generic_next_hook:nn.)
```

`__hook_try_declaring_generic_hook:nNNnn` now splits the hook name at the first / (if any) and first checks if it is a file-specific hook (they require some normalization) using `__hook_if_file_hook:wTF`. If not then check it is one of a predefined set for generic names. We also split off the second component to see if we have to make a reversed hook. In either case the function returns `true` for a generic hook and `false` in other cases.

```

400 <latexrelease>\cs_new_protected:Npn \_\_hook_try_declaring_generic_hook:nNNnn #1
401 <latexrelease> {
402     \_\_hook_if_file_hook:wTF #1 / \s_\_hook_mark
403     {
404         \exp_args:Ne \_\_hook_try_declaring_generic_hook_split:nNNnn
405         { \exp_args:Ne \_\_hook_file_hook_normalize:n {#1} }
406     }
407     { \_\_hook_try_declaring_generic_hook_split:nNNnn {#1} }
408 }

409 <latexrelease>\cs_new_protected:Npn \_\_hook_try_declaring_generic_hook_split:nNNnn #1 #2 #3
410 <latexrelease> {
411     \_\_hook_try_declaring_generic_hook:wnTF #1 // \scan_stop: {#1}
```

```

412 \langle latexrelease\rangle { #2 }
413 \langle latexrelease\rangle { #3 } {#1}
414 \langle latexrelease\rangle }
415 \langle latexrelease\rangle \EndIncludeInRelease

(End definition for \_\_hook\_try\_declaring\_generic\_hook:nNnn and
\_\_hook\_try\_declaring\_generic\_hook\_split:nNnn.)
```

__hook_try_declaring_generic_hook:wnTF

```

416 \langle latexrelease\rangle \IncludeInRelease{2021/11/15}{\_\_hook\_try\_declaring\_generic\_hook:wn}%
417 \langle latexrelease\rangle {Standardise-generic~hook~names}
418 \prg_new_protected_conditional:Npn \_\_hook_try_declaring_generic_hook:wn
419 #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
420 {
421   \_\_hook_if_generic:nTF {#5}
422   {
423     \_\_hook_if_usable:nF {#5}
424     {
```

If the hook doesn't exist yet we check if it is a cmd hook and if so we attempt patching the command in addition to declaring the hook.

For some commands this will not be possible, in which case __hook_patch_cmd_or_delay:Nnn (defined in ltcmdhooks) will generate an appropriate error message.

```

425 \str_if_eq:nnT {#1} { cmd }
426   { \_\_hook_try_put_cmd_hook:n {#5} }
```

Declare the hook always even if it can't really be used (error message generated elsewhere).

Here we use __hook_make_usable:n, so that a \hook_new:n is still possible later.

```

427   \_\_hook_make_usable:n {#5}
428   }
429   \_\_hook_if_generic_reversed:nT {#5}
430   { \tl_gset:cn { g\_hook\_#5\_reversed_tl } { - } }
431   \prg_return_true:
432 }
433 {
```

Generic hooks are all named $\langle type \rangle / \langle name \rangle / \langle place \rangle$, where $\langle type \rangle$ and $\langle place \rangle$ are predefined (\c_hook_generic_<type>/./<place>_tl), and $\langle name \rangle$ is the variable component. Older releases had some hooks with the $\langle name \rangle$ in the third part, so the code below supports that syntax for a while, with a warning.

The \exp_after:wN ... \exp:w trick is there to remove the conditional structure inserted by __hook_try_declaring_generic_hook:wnTF and thus allow access to the tokens that follow it, as is needed to keep things going.

When the deprecation cycle ends, the lines below should all be replaced by \prg_return_false::

```

434   \_\_hook_if_deprecated_generic:nTF {#5}
435   {
436     \_\_hook_deprecated_generic_warn:n {#5}
437     \exp_after:wN \_\_hook_declare_deprecated_generic:NNn
438     \exp:w % \exp_end:
439   }
440   { \prg_return_false: }
441 }
442 }
```

`__hook_DEPRECATED_GENERIC_WARN:Nn` will issue a deprecation warning for a given hook, and mark that hook such that the warning will not be issued again (multiple warnings can be issued, but only once per hook).

```

443 \cs_new_protected:Npn \__hook_DEPRECATED_GENERIC_WARN:n #1
444   { \__hook_DEPRECATED_GENERIC_WARN:w #1 \s__hook_mark }
445 \cs_new_protected:Npn \__hook_DEPRECATED_GENERIC_WARN:w
446   #1 / #2 / #3 \s__hook_mark
447   {
448     \ifcsexist:w __hook~#1/#2/#3 \cs_end: \else:
449       \msg_warning:nnnn { hooks } { generic-deprecated } {#1} {#2} {#3}
450     \fi:
451     \cs_gset_eq:cN { __hook~#1/#2/#3 } \scan_stop:
452   }

```

Now that the user has been told about the deprecation, we proceed by swapping *<name>* and *<place>* and adding the code to the correct hook.

```

453 \cs_new_protected:Npn \__hook_DO_DEPRECATED_GENERIC:Nn #1 #2
454   { \__hook_DO_DEPRECATED_GENERIC:Nw #1 #2 \s__hook_mark }
455 \cs_new_protected:Npn \__hook_DO_DEPRECATED_GENERIC:Nw #1
456   #2 / #3 / #4 \s__hook_mark
457   { #1 { #2 / #4 / #3 } }
458 \cs_new_protected:Npn \__hook_DECLARE_DEPRECATED_GENERIC>NNn #1 #2 #3
459   { \__hook_DECLARE_DEPRECATED_GENERIC>NNw #1 #2 #3 \s__hook_mark }
460 \cs_new_protected:Npn \__hook_DECLARE_DEPRECATED_GENERIC>NNw #1 #2
461   #3 / #4 / #5 \s__hook_mark
462   {
463     \__hook_TRY_DECLARING_GENERIC_HOOK:wnTF #3 / #5 / #4 / \scan_stop:
464     { #3 / #5 / #4 }
465     #1 #2 { #3 / #5 / #4 }
466   }
467 \end{IncludeInRelease}

468 \begin{IncludeInRelease}[2021/06/01]{\__hook_TRY_DECLARING_GENERIC_HOOK:wn}
469 \begin{IncludeInRelease}[Support~cmd~hooks]
470 \prg_new_protected_conditional:Npnn \__hook_TRY_DECLARING_GENERIC_HOOK:wn
471   #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
472 \begin{IncludeInRelease}
473   \tl_if_empty:nTF {#2}
474   { \prg_return_false: }
475 \begin{IncludeInRelease}
476   \prop_if_in:NnTF \c__hook_generics_prop {#1}
477   {
478     \__hook_if_usable:nF {#5}
479     {
480       \str_if_eq:nnT {#1} { cmd }
481       { \__hook_try_put_cmd_hook:n {#5} }
482       \__hook_make_usable:n {#5}
483     }
484   \prop_if_in:NnTF \c__hook_generics_reversed_ii_prop {#2}
485   { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
486   {
487     \prop_if_in:NnTF \c__hook_generics_reversed_iii_prop {#3}
488     { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
489   }

```

```

490 〈latexrelease〉          \prg_return_true:
491 〈latexrelease〉          }
492 〈latexrelease〉          { \prg_return_false: }
493 〈latexrelease〉          }
494 〈latexrelease〉          }
495 〈latexrelease〉\EndIncludeInRelease

496 〈latexrelease〉\IncludeInRelease{2020/10/01}{\_\_hook_try_declaring_generic_hook:wn}%
497 〈latexrelease〉          {Support-cmd-hooks}
498 〈latexrelease〉\prg_new_protected_conditional:Npnn \_\_hook_try_declaring_generic_hook:wn
499 〈latexrelease〉          #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
500 〈latexrelease〉          {
501 〈latexrelease〉          \tl_if_empty:nTF {#2}
502 〈latexrelease〉          { \prg_return_false: }
503 〈latexrelease〉          {
504 〈latexrelease〉          \prop_if_in:NnTF \c__hook_genetics_prop {#1}
505 〈latexrelease〉          {
506 〈latexrelease〉          \_\_hook_if_declared:nF {#5} { \hook_new:n {#5} }
507 〈latexrelease〉          \prop_if_in:NnTF \c__hook_genetics_reversed_ii_prop {#2}
508 〈latexrelease〉          { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
509 〈latexrelease〉          {
510 〈latexrelease〉          \prop_if_in:NnT \c__hook_genetics_reversed_iii_prop {#3}
511 〈latexrelease〉          { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
512 〈latexrelease〉          }
513 〈latexrelease〉          \prg_return_true:
514 〈latexrelease〉          }
515 〈latexrelease〉          { \prg_return_false: }
516 〈latexrelease〉          }
517 〈latexrelease〉          }
518 〈latexrelease〉
519 〈latexrelease〉\EndIncludeInRelease

```

(End definition for `__hook_try_declaring_generic_hook:wnTF` and others.)

`__hook_if_file_hook:p:w` `__hook_if_file_hook:wTF` `__hook_if_file_hook:wTF` checks if the argument is a valid file-specific hook (not, for example, `file/before`, but `file/foo.tex/before`). If it is a file-specific hook, then it executes the `<true>` branch, otherwise `<false>`.

```

520 〈latexrelease〉\IncludeInRelease{2021/11/15}{\_\_hook_if_file_hook:w}%
521 〈latexrelease〉          {Standardise-generic-hook-names}
522 \prg_new_conditional:Npnn \_\_hook_if_file_hook:w
523  #1 / #2 \s__hook_mark #3 { TF }
524  {
525  \_\_hook_if_generic:nTF {#3}
526  {
527  \str_if_eq:nnTF {#1} { file }
528  { \prg_return_true: }
529  { \prg_return_false: }
530  }
531  { \prg_return_false: }
532  }
533 〈latexrelease〉\EndIncludeInRelease

534 〈latexrelease〉\IncludeInRelease{2020/10/01}{\_\_hook_if_file_hook:w}%
535 〈latexrelease〉          {Standardise-generic-hook-names}
536 \prg_new_conditional:Npnn \_\_hook_if_file_hook:w

```

```

537 <latexrelease>      #1 / #2 / #3 \s__hook_mark { TF }
538 <latexrelease>  {
539 <latexrelease>    \str_if_eq:nnTF {#1} { file }
540 <latexrelease>    {
541 <latexrelease>      \bool_lazy_or:nnTF
542 <latexrelease>        { \tl_if_empty_p:n {#3} }
543 <latexrelease>        { \str_if_eq_p:nn {#3} { / } }
544 <latexrelease>        { \prg_return_false: }
545 <latexrelease>        {
546 <latexrelease>          \prop_if_in:NnTF \c__hook_generics_file_prop {#2}
547 <latexrelease>            { \prg_return_true: }
548 <latexrelease>            { \prg_return_false: }
549 <latexrelease>        }
550 <latexrelease>      }
551 <latexrelease>    { \prg_return_false: }
552 <latexrelease>  }
553 <latexrelease>\EndIncludeInRelease

```

(End definition for `_hook_if_file_hook:wTF.`)

```

\_\_hook\_file\_hook\_normalize:n
\_\_hook\_strip\_double\_slash:n
\_\_hook\_strip\_double\_slash:w

```

```

554 <latexrelease>\IncludeInRelease{2021/11/15}{\_\_hook\_file\_hook\_normalize:n}%
555 <latexrelease>                                {Standardise-generic-hook-names}
556 <latexrelease>\EndIncludeInRelease

```

When a file-specific hook is found, before being declared it is lightly normalized by `_hook_file_hook_normalize:n`. The current implementation just replaces two consecutive slashes (//) by a single one, to cope with simple cases where the user did something like `\def\input@path{./mypath/}`, in which case a hook would have to be `\AddToHook{file./mypath//file.tex/after}`.

```

557 <latexrelease>\IncludeInRelease{2020/10/01}{\_\_hook\_file\_hook\_normalize:n}%
558 <latexrelease>                                {Standardise-generic-hook-names}
559 <latexrelease>\cs_new:Npn \_\_hook\_file\_hook\_normalize:n #1
560 <latexrelease>  { \_\_hook_strip_double_slash:n {#1} }
561 <latexrelease>\cs_new:Npn \_\_hook_strip_double_slash:n #1
562 <latexrelease>  { \_\_hook_strip_double_slash:w #1 // \s__hook_mark }

```

This function is always called after testing if the argument is a file hook with `_hook_if_file_hook:wTF`, so we can assume it has three parts (it is either `file/.../before` or `file/.../after`), so we use `#1/#2/#3 //` instead of just `#1 //` to prevent losing a slash if the file name is empty.

```

563 <latexrelease>\cs_new:Npn \_\_hook_strip_double_slash:w #1/#2/#3 // #4 \s__hook_mark
564 <latexrelease>  {
565 <latexrelease>    \tl_if_empty:nTF {#4}
566 <latexrelease>      { #1/#2/#3 }
567 <latexrelease>      { \_\_hook_strip_double_slash:w #1/#2/#3 / #4 \s__hook_mark }
568 <latexrelease>  }
569 <latexrelease>\EndIncludeInRelease

```

(End definition for `_hook_file_hook_normalize:n`, `_hook_strip_double_slash:n`, and `_hook_strip_double_slash:w`)

```

\c__hook_generic_cmd/.before_tl
\c__hook_generic_cmd/.after_tl

```

Token lists defining the possible generic hooks. We don't provide any user interface to this as this is meant to be static.

cmd The generic hooks used for commands.
env The generic hooks used in \begin and \end.
file, package, class, include The generic hooks used when loading a file

```

570 <latexrelease>\IncludeInRelease{2021/11/15}{\c_hook_generics_prop}%
571 <latexrelease>                                {Standardise-generic~hook~names}
572 \clist_map_inline:nn { cmd , env , file , package , class , include }
573 {
574     \tl_const:cn { c__hook_generic_#1./before_tl } { + }
575     \tl_const:cn { c__hook_generic_#1./after_tl } { - }
576 }
577 \tl_const:cn { c__hook_generic_env./begin_tl } { + }
578 \tl_const:cn { c__hook_generic_env./end_tl } { + }
579 \tl_const:cn { c__hook_generic_include./end_tl } { - }

```

Deprecated generic hooks:

```

580 \clist_map_inline:nn { file , package , class , include }
581 {
582     \tl_const:cn { c__hook_DEPRECATED_#1./before_tl } { }
583     \tl_const:cn { c__hook_DEPRECATED_#1./after_tl } { }
584 }
585 \tl_const:cn { c__hook_DEPRECATED_include./end_tl } { }
586 <latexrelease>\EndIncludeInRelease
587 <latexrelease>\IncludeInRelease{2020/10/01}{\c_hook_generics_prop}%
588 <latexrelease>                                {Standardise-generic~hook~names}
589 <latexrelease>\prop_const_from_keyval:Nn \c_hook_generics_prop
590 <latexrelease>      {cmd=,env=,file=,package=,class=,include=}
591 <latexrelease>\EndIncludeInRelease

```

(End definition for \c_hook_generic_cmd./before_tl and others.)

The following generic hooks are supposed to use reverse ordering (the **ii** and **iii** names are kept for the deprecation cycle):

```

592 <latexrelease>\IncludeInRelease{2021/11/15}{\c_hook_generics_reversed_ii_prop}%
593 <latexrelease>                                {Standardise-generic~hook~names}
594 <latexrelease>\EndIncludeInRelease
595 <latexrelease>\IncludeInRelease{2020/10/01}{\c_hook_generics_reversed_ii_prop}%
596 <latexrelease>                                {Standardise-generic~hook~names}
597 <latexrelease>\prop_const_from_keyval:Nn \c_hook_generics_reversed_ii_prop {after=,end=}
598 <latexrelease>\prop_const_from_keyval:Nn \c_hook_generics_reversed_iii_prop {after=}
599 <latexrelease>\prop_const_from_keyval:Nn \c_hook_generics_file_prop {before=,after=}
600 <latexrelease>\EndIncludeInRelease

```

(End definition for \c_hook_generics_reversed_ii_prop, \c_hook_generics_reversed_iii_prop, and \c_hook_generics_file_prop.)

\hook_gremove_code:nn With \hook_gremove_code:nn{\hook}{\label} any code for **\hook** stored under **\label** is removed.

```

601 \cs_new_protected:Npn \hook_gremove_code:nn #1 #2
602   { \__hook_normalize_hook_args:Nnn \__hook_gremove_code:nn {#1} {#2} }
603 \cs_new_protected:Npn \__hook_gremove_code:nn #1 #2
604   {

```

First check that the hook code pool exists. `__hook_if_usable:nTF` isn't used here because it should be possible to remove code from a hook before its defined (see section 2.1.8).

```
605     \__hook_if_structure_exist:nTF {#1}
606     {
```

Then remove the chunk and run `__hook_update_hook_code:n` so that the execution token list reflects the change if we are after `\begin{document}`.

If all code is to be removed, clear the code pool `\g__hook_<hook>_code_prop`, the top-level code `__hook_toplevel<hook>`, and the next-execution code `__hook_next<hook>`.

```
607     \str_if_eq:nnTF {#2} {*}
608     {
609         \prop_gclear:c { g__hook_#1_code_prop }
610         \__hook_tl_gclear:c { __hook_toplevel~#1 }
611         \__hook_tl_gclear:c { __hook_next~#1 }
612     }
613     {
```

If the label is `top-level` then clear the token list, as all code there is under the same label.

```
614     \str_if_eq:nnTF {#2} { top-level }
615     {
616         \__hook_tl_gclear:c { __hook_toplevel~#1 } }
617         \prop_gpop:cnNF { g__hook_#1_code_prop } {#2} \l__hook_return_tl
618         { \msg_warning:nnnn { hooks } { cannot-remove } {#1} {#2} }
619     }
620 }
```

Finally update the code, if the hook exists.

```
621     \__hook_if_usable:nT {#1}
622     {
623         \__hook_update_hook_code:n {#1} }
```

If the code pool for this hook doesn't exist, show a warning:

```
624     {
625         \__hook_if_deprecated_generic:nTF {#1}
626         {
627             \__hook_deprecated_generic_warn:n {#1}
628             \__hook_do_deprecated_generic:Nn \__hook_gremove_code:nn {#1} {#2}
629         }
630         { \msg_warning:nnnn { hooks } { cannot-remove } {#1} {#2} }
631     }
632 }
```

(End definition for `\hook_gremove_code:nn` and `__hook_gremove_code:nn`. This function is documented on page 193.)

`\g__hook_??_code_prop`
`__hook???`
`\g__hook_??_reversed_tl`

Initially these variables simply used an empty “label” name (not two question marks). This was a bit unfortunate, because then `l3doc` complains about `__` in the middle of a command name when trying to typeset the documentation. However using a “normal” name such as `default` has the disadvantage of that being not really distinguishable from a real hook name. I now have settled for `??` which needs some gymnastics to get it into the csname, but since this is used a lot, the code should be fast, so this is not done with `c` expansion in the code later on.

`__hook_??` isn't used, but it has to be defined to trick the code into thinking that `??` is actually a hook.

```
633 \prop_new:c {g__hook_??_code_prop}
634 \prop_new:c {_hook-??}
```

Default rules are always given in normal ordering (never in reversed ordering). If such a rule is applied to a reversed hook it behaves as if the rule is reversed (e.g., `after` becomes `before`) because those rules are applied first and then the order is reversed.

```
635 \tl_new:c {g__hook_??_reversed_tl}
```

(End definition for `\g__hook_??_code_prop`, `__hook-??`, and `\g__hook_??_reversed_tl`.)

4.7 Setting rules for hooks code

`\hook_gset_rule:nnnn` With `\hook_gset_rule:nnnn{<hook>}{<label1>}{}{<relation>}{<label2>}` a relation is defined between the two code labels for the given `<hook>`. The special hook `??` stands for *any* hook, which sets a default rule (to be used if no other relation between the two hooks exist).

```
636 \cs_new_protected:Npn \hook_gset_rule:nnnn #1#2#3#4
637 {
638     \__hook_normalize_hook_rule_args:Nnnnn \__hook_gset_rule:nnnn
639     {#1} {#2} {#3} {#4}
640 }
641 \cs_new_protected:Npn \__hook_gset_rule:nnnn #1#2#3#4
642 {
643     \__hook_if_DEPRECATED_GENERIC:nT {#1}
644     {
645         \__hook_DEPRECATED_GENERIC_WARN:n {#1}
646         \__hook_DO_DEPRECATED_GENERIC:Nn \__hook_gset_rule:nnnn {#1}
647         {#2} {#3} {#4}
648         \exp_after:wN \use_none:nnnnnnnn \use_none:n
649     }
650 }
```

First we ensure the basic data structure of the hook exists:

```
650 \__hook_init_structure:n {#1}
```

Then we clear any previous relationship between both labels.

```
651 \__hook_rule_gclear:nnn {#1} {#2} {#4}
```

Then we call the function to handle the given rule. Throw an error if the rule is invalid.

```
652 \cs_if_EXIST_USE:cTF { __hook_rule_#3_gset:nnn }
653 {
654     {#1} {#2} {#4}
655     \__hook_update_hook_code:n {#1}
656 }
657 { \msg_error:nnnnn { hooks } { unknown-rule }
658     {#1} {#2} {#3} {#4} }
659 }
```

(End definition for `\hook_gset_rule:nnnn` and `__hook_gset_rule:nnnn`. This function is documented on page 198.)

```
\_\_hook\_rule\_before\_gset:nnn
\_\_hook\_rule\_after\_gset:nnn
\_\_hook\_rule\_<\_gset:nnn
\_\_hook\_rule\_>\_gset:nnn
```

Then we add the new rule. We need to normalize the rules here to allow for faster processing later. Given a pair of labels l_A and l_B , the rule $l_A > l_B$ is the same as $l_B < l_A$ only presented differently. But by normalizing the forms of the rule to a single representation, say, $l_B < l_A$, reduces the time spent looking for the rules later considerably.

Here we do that normalization by using `\(pdf)strcmp` to lexically sort labels l_A and l_B to a fixed order. This order is then enforced every time these two labels are used together.

Here we use `__hook_label_pair:nn {⟨hook⟩} {⟨lA⟩} {⟨lB⟩}` to build a string $l_B | l_A$ with a fixed order, and use `__hook_label_ordered:nnTF` to apply the correct rule to the pair of labels, depending if it was sorted or not.

```
660 \cs_new_protected:Npn \_\_hook\_rule\_before\_gset:nnn #1#2#3
661 {
662     \_\_hook\_tl\_gset:cx { g\_hook\_#1\_rule\_ \_\_hook\_label\_pair:nn {#2} {#3} _tl }
663     { \_\_hook\_label\_ordered:nnTF {#2} {#3} { < } { > } }
664 }
665 \cs_new_eq:cN { \_\_hook\_rule\_<\_gset:nnn } \_\_hook\_rule\_before\_gset:nnn
666 \cs_new_protected:Npn \_\_hook\_rule\_after\_gset:nnn #1#2#3
667 {
668     \_\_hook\_tl\_gset:cx { g\_hook\_#1\_rule\_ \_\_hook\_label\_pair:nn {#3} {#2} _tl }
669     { \_\_hook\_label\_ordered:nnTF {#3} {#2} { < } { > } }
670 }
671 \cs_new_eq:cN { \_\_hook\_rule\_>\_gset:nnn } \_\_hook\_rule\_after\_gset:nnn
```

(End definition for `__hook_rule_before_gset:nnn` and others.)

```
\_\_hook\_rule\_voids\_gset:nnn
```

This rule removes (clears, actually) the code from label #3 if label #2 is in the hook #1.

```
672 \cs_new_protected:Npn \_\_hook\_rule\_voids\_gset:nnn #1#2#3
673 {
674     \_\_hook\_tl\_gset:cx { g\_hook\_#1\_rule\_ \_\_hook\_label\_pair:nn {#2} {#3} _tl }
675     { \_\_hook\_label\_ordered:nnTF {#2} {#3} { -> } { <- } }
676 }
```

(End definition for `__hook_rule_voids_gset:nnn`.)

```
\_\_hook\_rule\_incompatible-error\_gset:nnn
```

```
\_\_hook\_rule\_incompatible-warning\_gset:nnn
```

These relations make an error/warning if labels #2 and #3 appear together in hook #1.

```
677 \cs_new_protected:cpx { \_\_hook\_rule\_incompatible-error\_gset:nnn } #1#2#3
678 { \_\_hook\_tl\_gset:cn { g\_hook\_#1\_rule\_ \_\_hook\_label\_pair:nn {#2} {#3} _tl }
679     { xE } }
680 \cs_new_protected:cpx { \_\_hook\_rule\_incompatible-warning\_gset:nnn } #1#2#3
681 { \_\_hook\_tl\_gset:cn { g\_hook\_#1\_rule\_ \_\_hook\_label\_pair:nn {#2} {#3} _tl }
682     { xW } }
```

(End definition for `__hook_rule_incompatible-error_gset:nnn` and
`__hook_rule_incompatible-warning_gset:nnn`.)

```
\_\_hook\_rule\_unrelated\_gset:nnn
\_\_hook\_rule\_gclear:nnn
```

Undo a setting. `__hook_rule_unrelated_gset:nnn` doesn't need to do anything, since we use `__hook_rule_gclear:nnn` before setting any rule.

```
683 \cs_new_protected:Npn \_\_hook\_rule\_unrelated\_gset:nnn #1#2#3 { }
684 \cs_new_protected:Npn \_\_hook\_rule\_gclear:nnn #1#2#3
685 { \cs_undefine:c { g\_hook\_#1\_rule\_ \_\_hook\_label\_pair:nn {#2} {#3} _tl } }
```

(End definition for `__hook_rule_unrelated_gset:nnn` and `__hook_rule_gclear:nnn`.)

__hook_label_pair:nn Ensure that the lexically greater label comes first.

```
686 \cs_new:Npn \_\_hook_label_pair:nn #1#2
687 {
688     \if_case:w \_\_hook_str_compare:nn {#1} {#2} \exp_stop_f:
689         #1 | #1 % 0
690     \or: #1 | #2 % +1
691     \else: #2 | #1 % -1
692     \fi:
693 }
```

(End definition for __hook_label_pair:nn.)

__hook_label_ordered_p:nn Check that labels #1 and #2 are in the correct order (as returned by __hook_label_pair:nn) and if so return true, else return false.

```
694 \prg_new_conditional:Npnn \_\_hook_label_ordered:nn #1#2 { TF }
695 {
696     \if_int_compare:w \_\_hook_str_compare:nn {#1} {#2} > 0 \exp_stop_f:
697     \prg_return_true:
698     \else:
699     \prg_return_false:
700     \fi:
701 }
```

(End definition for __hook_label_ordered:nnTF.)

__hook_if_label_case:nnnn To avoid doing the string comparison twice in __hook_initialize_single:NNn (once with \str_if_eq:nn and again with __hook_label_ordered:nn), we use a three-way branching macro that will compare #1 and #2 and expand to \use_i:nn if they are equal, \use_ii:nn if #1 is lexically greater, and \use_iii:nn otherwise.

```
702 \cs_new:Npn \_\_hook_if_label_case:nnnn #1#2
703 {
704     \cs:w use_
705     \if_case:w \_\_hook_str_compare:nn {#1} {#2}
706         i \or: ii \else: iii \fi: :nnn
707     \cs_end:
708 }
```

(End definition for __hook_if_label_case:nnnn.)

__hook_update_hook_code:n Before \begin{document} this does nothing, in the body it reinitializes the hook code using the altered data.

```
709 \cs_new_eq:NN \_\_hook_update_hook_code:n \use_none:n
```

(End definition for __hook_update_hook_code:n.)

__hook_initialize_all: Initialize all known hooks (at \begin{document}), i.e., update the fast execution token lists to hold the necessary code in the right order.

```
710 \cs_new_protected:Npn \_\_hook_initialize_all: {
```

First we change __hook_update_hook_code:n which so far was a no-op to now initialize one hook. This way any later updates to the hook will run that code and also update the execution token list.

```
711     \cs_gset_eq:NN \_\_hook_update_hook_code:n \_\_hook_initialize_hook_code:n
```

Now we loop over all hooks that have been defined and update each of them.

```

712  \_\_hook\_debug:n { \prop_gclear:N \g\_hook\_used\_prop }
713  \seq_map_inline:Nn \g\_hook\_all\_seq
714  {
715      \_\_hook_update_hook_code:n {##1}
716  }

```

If we are debugging we show results hook by hook for all hooks that have data.

```

717  \_\_hook\_debug:n
718  { \iow_term:x{^^JAll~ initialized~ (non-empty)~ hooks:}
719      \prop_map_inline:Nn \g\_hook\_used\_prop
720      { \iow_term:x{^^J~ ##1~ ->~
721          \exp_not:v {_\_hook~##1}~ }
722      }
723  }

```

After all hooks are initialized we change the “use” to just call the hook code and not initialize it (as it was done in the preamble).

```

724  \cs_gset_eq:NN \hook_use:n \_\_hook_use_initialized:n
725  \cs_gset_eq:NN \_\_hook_preamble_hook:n \use_none:n
726 }

```

(End definition for __hook_initialize_all:.)

__hook_initialize_hook_code:n
Initializing or reinitializing the fast execution hook code. In the preamble this is selectively done in case a hook gets used and at \begin{document} this is done for all hooks and afterwards only if the hook code changes.

```

727  \cs_new_protected:Npn \_\_hook_initialize_hook_code:n #1
728  {
729      \_\_hook_debug:n{ \iow_term:x{^^JUpdate~ code~ for~ hook-
730                      '#1' \on@line :^^J} }

```

This does the sorting and the updates. First thing we do is to check if a legacy hook macro exists and if so we add it to the hook under the label `legacy`. This might make the hook non-empty so we have to do this before the then following test.

```
731  \_\_hook_include_legacy_code_chunk:n {#1}
```

If there aren’t any code chunks for the current hook, there is no point in even starting the sorting routine so we make a quick test for that and in that case just update __hook_{\hook} to hold the top-level and next code chunks. If there are code chunks we call __hook_initialize_single:NNn and pass to it ready made csnames as they are needed several times inside. This way we save a bit on processing time if we do that up front.

```

732  \_\_hook_if_usable:nT {#1}
733  {
734      \prop_if_empty:cTF { g\_hook_#1_code_prop }
735      {
736          \_\_hook_tl_gset:co { _hook~##1 }
737          {
738              \cs:w _hook_toplevel~##1 \exp_after:wN \cs_end:
739              \cs:w _hook_next~##1 \cs_end:
740          }
741      }
742  {

```

By default the algorithm sorts the code chunks and then saves the result in a token list for fast execution; this is done by adding the code chunks one after another, using `\tl_gput_right:N`. When we sort code for a reversed hook, all we have to do is to add the code chunks in the opposite order into the token list. So all we have to do in preparation is to change two definitions that are used later on.

```

743     \__hook_if_reversed:nTF {#1}
744     { \cs_set_eq:NN \__hook_tl_gput:Nn      \__hook_tl_gput_left:Nn
745       \cs_set_eq:NN \__hook_clist_gput:NV \clist_gput_left:NV }
746     { \cs_set_eq:NN \__hook_tl_gput:Nn      \__hook_tl_gput_right:Nn
747       \cs_set_eq:NN \__hook_clist_gput:NV \clist_gput_right:NV }

```

When sorting, some relations (namely `voids`) need to act destructively on the code property lists to remove code that shouldn't appear in the sorted hook token list, so we make a copy of the code property list that we can safely work on without changing the main one.

```

748     \prop_set_eq:Nc \l__hook_work_prop { g__hook:#1_code_prop }
749     \__hook_initialize_single:ccn
750     { __hook~#1 } { g__hook:#1_labels_clist } {#1}

```

For debug display we want to keep track of those hooks that actually got code added to them, so we record that in plist. We use a plist to ensure that we record each hook name only once, i.e., we are only interested in storing the keys and the value is arbitrary.

```

751     \__hook_debug:n{ \exp_args:NNx \prop_gput:Nnn
752                     \g__hook_used_prop {#1}{ } }
753   }
754 }
755 }
```

(End definition for `__hook_initialize_hook_code:n`.)

`__hook_tl_csname:n`
`__hook_seq_csname:n` It is faster to pass a single token and expand it when necessary than to pass a bunch of character tokens around.

FMi: note to myself: verify

```

756 \cs_new:Npn \__hook_tl_csname:n #1 { l__hook_label_#1_tl }
757 \cs_new:Npn \__hook_seq_csname:n #1 { l__hook_label_#1_seq }

```

(End definition for `__hook_tl_csname:n` and `__hook_seq_csname:n`.)

`\l__hook_labels_seq`
`\l__hook_labels_int`
`\l__hook_front_tl`
`\l__hook_rear_tl`
`\l__hook_label_0_tl` For the sorting I am basically implementing Knuth's algorithm for topological sorting as given in TAOCP volume 1 pages 263–266. For this algorithm we need a number of local variables:

- List of labels used in the current hook to label code chunks:

```
758 \seq_new:N \l__hook_labels_seq
```

- Number of labels used in the current hook. In Knuth's algorithm this is called N :

```
759 \int_new:N \l__hook_labels_int
```

- The sorted code list to be build is managed using two pointers one to the front of the queue and one to the rear. We model this using token list pointers. Knuth calls them F and R :

```
760   \tl_new:N \l__hook_front_tl
761   \tl_new:N \l__hook_rear_tl
```

- The data for the start of the queue is kept in this token list, it corresponds to what Don calls $\text{QLINK}[0]$ but since we aren't manipulating individual words in memory it is slightly differently done:

```
762   \tl_new:c { \__hook_tl_cname:n { 0 } }
```

(End definition for `\l__hook_labels_seq` and others.)

`__hook_initialize_single:Nn` implements the sorting of the code chunks for a hook and saves the result in the token list for fast execution (#4). The arguments are $\langle\text{hook-code-plist}\rangle$, $\langle\text{hook-code-tl}\rangle$, $\langle\text{hook-top-level-code-tl}\rangle$, $\langle\text{hook-next-code-tl}\rangle$, $\langle\text{hook-ordered-labels-clist}\rangle$ and $\langle\text{hook-name}\rangle$ (the latter is only used for debugging—the $\langle\text{hook-rule-plist}\rangle$ is accessed using the $\langle\text{hook-name}\rangle$).

The additional complexity compared to Don's algorithm is that we do not use simple positive integers but have arbitrary alphanumeric labels. As usual Don's data structures are chosen in a way that one can omit a lot of tests and I have mimicked that as far as possible. The result is a restriction I do not test for at the moment: a label can't be equal to the number 0!

FMi: Needs checking for, just in case ... maybe

```
763 \cs_new_protected:Npn \__hook_initialize_single:Nn #1#2#3
764 {
```

Step T1: Initialize the data structure ...

```
765   \seq_clear:N \l__hook_labels_seq
766   \int_zero:N \l__hook_labels_int
```

Store the name of the hook:

```
767   \tl_set:Nn \l__hook_cur_hook_tl {\#3}
```

We loop over the property list holding the code and record all the labels listed there. Only the rules for those labels are of interest to us. While we are at it we count them (which gives us the N in Knuth's algorithm). The prefix `label_` is added to the variables to ensure that labels named `front`, `rear`, `labels`, or `return` don't interact with our code.

```
768   \prop_map_inline:Nn \l__hook_work_prop
769   {
770     \int_incr:N \l__hook_labels_int
771     \seq_put_right:Nn \l__hook_labels_seq {\##1}
772     \__hook_tl_set:cn { \__hook_tl_cname:n {\##1} } { 0 }
773     \seq_clear_new:c { \__hook_seq_cname:n {\##1} }
774 }
```

Steps T2 and T3: Here we sort the relevant rules into the data structure...

This loop constitutes a square matrix of the labels in `\l__hook_work_prop` in the vertical and the horizontal directions. However, since the rule $l_A \langle rel \rangle l_B$ is the same as $l_B \langle rel \rangle^{-1} l_A$ we can cut the loop short at the diagonal of the matrix (*i.e.*, when both labels are equal), saving a good amount of time. The way the rules were set up (see

the implementation of `__hook_rule_before_gset:nnn` above) ensures that we have no rule in the ignored side of the matrix, and all rules are seen. The rules are applied in `__hook_apply_label_pair:nnn`, which takes the properly-ordered pair of labels as argument.

```

775   \prop_map_inline:Nn \l__hook_work_prop
776   {
777     \prop_map_inline:Nn \l__hook_work_prop
778     {
779       \_\_hook_if_label_case:nnnn {##1} {####1}
780       { \prop_map_break: }
781       { \_\_hook_apply_label_pair:nnn {##1} {####1} }
782       { \_\_hook_apply_label_pair:nnn {####1} {##1} }
783       {#3}
784     }
785   }

```

Now take a breath, and look at the data structures that have been set up:

```

786   \_\_hook_debug:n { \_\_hook_debug_label_data:N \l__hook_work_prop }
Step T4:

```

```

787   \tl_set:Nn \l__hook_rear_tl { 0 }
788   \tl_set:cn { \_\_hook_tl_cname:n { 0 } } { 0 }
789   \seq_map_inline:Nn \l__hook_labels_seq
790   {
791     \int_compare:nNnT { \cs:w \_\_hook_tl_cname:n {##1} \cs_end: } = 0
792     {
793       \tl_set:cn { \_\_hook_tl_cname:n { \l__hook_rear_tl } }{##1}
794       \tl_set:Nn \l__hook_rear_tl {##1}
795     }
796   }
797   \tl_set_eq:Nc \l__hook_front_tl { \_\_hook_tl_cname:n { 0 } }
798   \_\_hook_tl_gclear:N #1
799   \clist_gclear:N #2

```

The whole loop gets combined in steps T5–T7:

```

800   \bool_while_do:nn { ! \str_if_eq_p:Vn \l__hook_front_tl { 0 } }
801   {

```

This part is step T5:

```

802   \int_decr:N \l__hook_labels_int
803   \prop_get:NVN \l__hook_work_prop \l__hook_front_tl \l__hook_return_tl
804   \exp_args:NNV \_\_hook_tl_gput:Nn #1 \l__hook_return_tl
805   \_\_hook_clist_gput:NV #2 \l__hook_front_tl
806   \_\_hook_debug:n{ \iow_term:x{Handled~ code~ for~ \l__hook_front_tl} }

```

This is step T6, except that we don't use a pointer P to move through the successors, but instead use $\#\#1$ of the mapping function.

```

807   \seq_map_inline:cn { \_\_hook_seq_cname:n { \l__hook_front_tl } }
808   {
809     \tl_set:cx { \_\_hook_tl_cname:n {##1} }
810     { \int_eval:n
811       { \cs:w \_\_hook_tl_cname:n {##1} \cs_end: - 1 }
812     }
813     \int_compare:nNnT

```

```

814     { \cs:w \__hook_tl_cname:n {##1} \cs_end: } = 0
815     {
816         \tl_set:cn { \__hook_tl_cname:n { \l__hook_rear_tl } } {##1}
817         \tl_set:Nn \l__hook_rear_tl
818             {##1}
819     }

```

and here is step T7:

```

820         \tl_set_eq:Nc \l__hook_front_tl
821             { \__hook_tl_cname:n { \l__hook_front_tl } }

```

This is step T8: If we haven't moved the code for all labels (i.e., if `\l__hook_labels_int` is still greater than zero) we have a loop and our partial order can't be flattened out.

```

822     }
823     \int_compare:nNnf \l__hook_labels_int = 0
824     {
825         \iow_term:x{=====}
826         \iow_term:x{Error:~ label~ rules~ are~ incompatible:}

```

This is not really the information one needs in the error case but it will do for now
...

FMi: improve output on a rainy day

```

827         \__hook_debug_label_data:N \l__hook_work_prop
828             \iow_term:x{=====}
829     }

```

After we have added all hook code to #1, we finish it off by adding extra code for the top-level (#2) and for one time execution (#3). These should normally be empty. The top-level code is added with `__hook_tl_gput:Nn` as that might change for a reversed hook (then top-level is the very first code chunk added). The next code is always added last.

```

830     \exp_args:NNo \__hook_tl_gput:Nn #1 { \cs:w __hook_toplevel~#3 \cs_end: }
831         \__hook_tl_gput_right:No #1 { \cs:w __hook_next~#3 \cs_end: }
832     }
833 \cs_generate_variant:Nn \__hook_initialize_single:NNn { cc }

(End definition for \__hook_initialize_single:NNn.)

```

`__hook_tl_gput:Nn`
`__hook_clist_gput:NV` These append either on the right (normal hook) or on the left (reversed hook). This is setup up in `__hook_initialize_hook_code:n`, elsewhere their behavior is undefined.

```

834 \cs_new:Npn \__hook_tl_gput:Nn { \ERROR }
835 \cs_new:Npn \__hook_clist_gput:NV { \ERROR }

```

(End definition for `__hook_tl_gput:Nn` and `__hook_clist_gput:NV`.)

`__hook_apply_label_pair:nnn`
`__hook_label_if_exist_apply:nnmf` This is the payload of steps T2 and T3 executed in the loop described above. This macro assumes #1 and #2 are ordered, which means that any rule pertaining the pair #1 and #2 is $\text{\g}_{\text{\hook}}\langle\text{\hook}\rangle_{\text{\rule}}\#1|\#2_t1$, and not $\text{\g}_{\text{\hook}}\langle\text{\hook}\rangle_{\text{\rule}}\#2|\#1_t1$. This also saves a great deal of time since we only need to check the order of the labels once.

The arguments here are $\langle\text{\label1}\rangle$, $\langle\text{\label2}\rangle$, $\langle\text{\hook}\rangle$, and $\langle\text{\hook-code-plist}\rangle$. We are about to apply the next rule and enter it into the data structure. `__hook_apply_label_pair:nnn` will just call `__hook_label_if_exist_apply:nnmf` for the $\langle\text{\hook}\rangle$, and if no rule is found, also try the $\langle\text{\hook}\rangle$ name ?? denoting a default hook rule.

`__hook_label_if_exist_apply:nnnF` will check if the rule exists for the given hook, and if so call `__hook_apply_rule:nnn`.

```
836 \cs_new_protected:Npn \__hook_apply_label_pair:nnn #1#2#3
837 {
```

Extra complication: as we use default rules and local hook specific rules we first have to check if there is a local rule and if that exist use it. Otherwise check if there is a default rule and use that.

```
838     \__hook_label_if_exist_apply:nnnF {#1} {#2} {#3}
839 {
```

If there is no hook-specific rule we check for a default one and use that if it exists.

```
840         \__hook_label_if_exist_apply:nnnF {#1} {#2} { ?? } { }
841     }
842 }
843 \cs_new_protected:Npn \__hook_label_if_exist_apply:nnnF #1#2#3
844 {
845     \if cs_exist:w g__hook_ #3 _rule_ #1 | #2 _tl \cs_end:
```

What to do precisely depends on the type of rule we have encountered. If it is a `before` rule it will be handled by the algorithm but other types need to be managed differently. All this is done in `__hook_apply_rule:nnnN`.

```
846     \__hook_apply_rule:nnn {#1} {#2} {#3}
847     \exp_after:wN \use_none:n
848 \else:
849     \use:nn
850 \fi:
851 }
```

(End definition for `__hook_apply_label_pair:nnn` and `__hook_label_if_exist_apply:nnnF`.)

`__hook_apply_rule:nnn` This is the code executed in steps T2 and T3 while looping through the matrix. This is part of step T3. We are about to apply the next rule and enter it into the data structure. The arguments are `<label1>`, `<label2>`, `<hook-name>`, and `<hook-code-plist>`.

```
852 \cs_new_protected:Npn \__hook_apply_rule:nnn #1#2#3
853 {
854     \cs:w __hook_apply_
855     \cs:w g__hook_#3_reversed_tl \cs_end: rule_
856     \cs:w g__hook_ #3 _rule_ #1 | #2 _tl \cs_end: :nnn \cs_end:
857     {#1} {#2} {#3}
858 }
```

(End definition for `__hook_apply_rule:nnn`.)

`__hook_apply_rule_<:nnn` The most common cases are `<` and `>` so we handle that first. They are relations \prec and \succ in TAOCP, and they dictate sorting.

```
859 \cs_new_protected:cpn { __hook_apply_rule_<:nnn } #1#2#3
860 {
861     \__hook_debug:n { \__hook_msg_pair_found:nnn {#1} {#2} {#3} }
862     \tl_set:cx { \__hook_tl_cname:n {#2} }
863     { \int_eval:n{ \cs:w \__hook_tl_cname:n {#2} \cs_end: + 1 } }
864     \seq_put_right:cn{ \__hook_seq_cname:n {#1} }{#2}
865 }
866 \cs_new_protected:cpn { __hook_apply_rule_>:nnn } #1#2#3
867 {
```

```

868     \__hook_debug:n { \__hook_msg_pair_found:nnn {#1} {#2} {#3} }
869     \tl_set:cx { \__hook_tl_cname:n {#1} }
870         { \int_eval:n{ \cs:w \__hook_tl_cname:n {#1} \cs_end: + 1 } }
871     \seq_put_right:cn{ \__hook_seq_cname:n {#2} }{#1}
872 }
```

(End definition for `__hook_apply_rule_<:nnn` and `__hook_apply_rule_>:nnn`.)

`__hook_apply_rule_xE:nnn` These relations make two labels incompatible within a hook. `xE` makes raises an error if the labels are found in the same hook, and `xW` makes it a warning.

```

873 \cs_new_protected:cpn { __hook_apply_rule_xE:nnn } #1#2#3
874 {
875     \__hook_debug:n { \__hook_msg_pair_found:nnn {#1} {#2} {#3} }
876     \msg_error:nnnnn { hooks } { labels-incompatible }
877         {#1} {#2} {#3} { 1 }
878     \use:c { __hook_apply_rule_->:nnn } {#1} {#2} {#3}
879     \use:c { __hook_apply_rule_-<:nnn } {#1} {#2} {#3}
880 }
881 \cs_new_protected:cpn { __hook_apply_rule_xW:nnn } #1#2#3
882 {
883     \__hook_debug:n { \__hook_msg_pair_found:nnn {#1} {#2} {#3} }
884     \msg_warning:nnnnn { hooks } { labels-incompatible }
885         {#1} {#2} {#3} { 0 }
886 }
```

(End definition for `__hook_apply_rule_xE:nnn` and `__hook_apply_rule_xW:nnn`.)

`__hook_apply_rule_->:nnn` `__hook_apply_rule_-<:nnn` If we see `->` we have to drop code for label `#3` and carry on. We could do a little better and drop everything for that label since it doesn't matter where we put such empty code. However that would complicate the algorithm a lot with little gain.¹⁰ So we still unnecessarily try to sort it in and depending on the rules that might result in a loop that is otherwise resolved. If that turns out to be a real issue, we can improve the code.

Here the code is removed from `\l__hook_cur_hook_tl` rather than `#3` because the latter may be `??`, and the default hook doesn't store any code. Removing it instead from `\l__hook_cur_hook_tl` makes the default rules `->` and `-<` work properly.

```

887 \cs_new_protected:cpn { __hook_apply_rule_->:nnn } #1#2#3
888 {
889     \__hook_debug:n
890     {
891         \__hook_msg_pair_found:nnn {#1} {#2} {#3}
892         \iow_term:x{---~ Drop~ '#2'~ code~ from~
893             \iow_char:N \\ g_hook_\l__hook_cur_hook_tl _code_prop ~
894             because~ of~ '#1' }
895     }
896     \prop_put:Nnn \l__hook_work_prop {#2} { }
897 }
898 \cs_new_protected:cpn { __hook_apply_rule_-<:nnn } #1#2#3
899 {
900     \__hook_debug:n
901     {
902         \__hook_msg_pair_found:nnn {#1} {#2} {#3}
```

¹⁰This also has the advantage that the result of the sorting doesn't change, as it might otherwise do (for unrelated chunks) if we aren't careful.

```

903         \iow_term:x{--->~ Drop~ '#1'~ code~ from~
904             \iow_char:N \\ g__hook_ \l__hook_cur_hook_tl _code_prop ~
905             because~ of~ '#2' }
906     }
907     \prop_put:Nnn \l__hook_work_prop {#1} { }
908 }

```

(End definition for `_hook_apply_rule_->:nnn` and `_hook_apply_rule_-<:nnn`.)

`_hook_apply_-rule_-<:nnn` Reversed rules.

```

909 \cs_new_eq:cc { __hook_apply_-rule_-<:nnn } { __hook_apply_rule_->:nnn }
910 \cs_new_eq:cc { __hook_apply_-rule_->:nnn } { __hook_apply_rule_-<:nnn }
911 \cs_new_eq:cc { __hook_apply_-rule_-<-:nnn } { __hook_apply_rule_-<-:nnn }
912 \cs_new_eq:cc { __hook_apply_-rule_->:nnn } { __hook_apply_rule_->:nnn }
913 \cs_new_eq:cc { __hook_apply_-rule_xE:nnn } { __hook_apply_rule_xE:nnn }
914 \cs_new_eq:cc { __hook_apply_-rule_xW:nnn } { __hook_apply_rule_xW:nnn }

```

(End definition for `_hook_apply_-rule_-<:nnn` and others.)

`_hook_msg_pair_found:nnn` A macro to avoid moving this many tokens around.

```

915 \cs_new_protected:Npn \_hook_msg_pair_found:nnn #1#2#3
916 {
917     \iow_term:x{~ \str_if_eq:nnTF {#3} {??} {default} {~normal} ~
918         rule~ \_hook_label_pair:nn {#1} {#2}:~}
919         \use:c { g__hook_#3_rule_ \_hook_label_pair:nn {#1} {#2} _tl } ~
920         found}
921 }

```

(End definition for `_hook_msg_pair_found:nnn`.)

`_hook_debug_label_data:N`

```

922 \cs_new_protected:Npn \_hook_debug_label_data:N #1 {
923     \iow_term:x{Code~ labels~ for~ sorting:}
924     \iow_term:x{~ \seq_use:Nnnn\l__hook_labels_seq {-and-}{,~}{-and-} }
925     \iow_term:x{^^J Data~ structure~ for~ label~ rules:}
926     \prop_map_inline:Nn #1
927     {
928         \iow_term:x{~ ##1~ == \tl_use:c{ \_hook_tl_cname:n {##1} }~ -~}
929         \seq_use:cnnn{ \_hook_seq_cname:n {##1} }{~-~}{-->~}{~-~}
930     }
931 }
932 \iow_term:x{}
933 }

```

(End definition for `_hook_debug_label_data:N`.)

`\hook_show:n`
`\hook_log:n`

This writes out information about the hook given in its argument onto the `.log` file and the terminal, if `\show_hook:n` is used. Internally both share the same structure, except that at the end, `\hook_show:n` triggers TeX's prompt.

```

934 \cs_new_protected:Npn \hook_log:n #1
935 {
936     \cs_set_eq:NN \_hook_log_cmd:x \iow_log:x
937     \_hook_normalize_hook_args:Nn \_hook_log:nN {#1} \tl_log:x
938 }
939 \cs_new_protected:Npn \hook_show:n #1

```

```

940  {
941    \cs_set_eq:NN \__hook_log_cmd:x \iow_term:x
942    \__hook_normalize_hook_args:Nn \__hook_log:nN {#1} \tl_show:x
943  }
944 \cs_new_protected:Npn \__hook_log_line:x #1
945  { \__hook_log_cmd:x { >~#1 } }
946 \cs_new_protected:Npn \__hook_log_line_indent:x #1
947  { \__hook_log_cmd:x { >~\@spaces #1 } }

948 \cs_new_protected:Npn \__hook_log:nN #1 #2
949  {
950    \__hook_if_deprecated_generic:nT {#1}
951    {
952      \__hook_DEPRECATED_GENERIC_WARN:n {#1}
953      \__hook_do_DEPRECATED_GENERIC:Nn \__hook_log:nN {#1} #2
954      \exp_after:wN \use:none:nnnnnnnn \use:none:nnnn
955    }
956    \__hook_preamble_hook:n {#1}
957    \__hook_log_cmd:x
958    { ^~J ->~The~ \__hook_if_generic:nT {#1} { generic~ } hook~'#1': }

959    \__hook_if_usable:nF {#1}
960    { \__hook_log_line:x { The~hook~is~not~declared. } }
961    \__hook_if_disabled:nT {#1}
962    { \__hook_log_line:x { The~hook~is~disabled. } }
963    \hook_if_empty:nTF {#1}
964    { #2 { The~hook~is~empty } }
965    {
966      \__hook_log_line:x { Code~chunks: }
967      \prop_if_empty:cTF { g__hook_#1_code_prop }
968      { \__hook_log_line_indent:x { --- } }
969      {
970        \prop_map_inline:cn { g__hook_#1_code_prop }
971        { \__hook_log_line_indent:x { ##1~->-\tl_to_str:n {##2} } }
972      }

```

If there is code in the top-level token list, print it:

```

973  \__hook_log_line:x
974  {
975    Document-level~(top-level)~code
976    \__hook_if_usable:nT {#1}
977    { ~(executed~\__hook_if_reversed:nTF {#1} {first} {last} ) : }
978  }
979  \__hook_log_line_indent:x
980  {
981    \tl_if_empty:cTF { __hook_toplevel~#1 }
982    { --- }
983    { -> ~ \exp_args:Nv \tl_to_str:n { __hook_toplevel~#1 } }
984  }

985  \__hook_log_line:x { Extra~code~for~next~invocation: }
986  \__hook_log_line_indent:x
987  {
988    \tl_if_empty:cTF { __hook_next~#1 }
989    { --- }

```

If the token list is not empty we want to display it but without the first tokens (the code to clear itself) so we call a helper command to get rid of them.

```
990      { ->~ \exp_args:Nv \__hook_log_next_code:n { __hook_next~#1 } }
991      }
```

Loop through the rules in a hook and for every rule found, print it. If no rule is there, print ---. The boolean \l__hook_tmpa_bool here indicates if the hook has no rules.

```
992      \__hook_log_line:x { Rules: }
993      \bool_set_true:N \l__hook_tmpa_bool
994      \__hook_list_rules:nn {#1}
995      {
996          \bool_set_false:N \l__hook_tmpa_bool
997          \__hook_log_line_indent:x
998          {
999              ##2~ with~
1000              \str_if_eq:nnT {##3} {??} { default~ }
1001                  relation~ ##1
1002              }
1003          }
1004          \bool_if:NT \l__hook_tmpa_bool
1005          { \__hook_log_line_indent:x { --- } }
```

When the hook is declared (that is, the sorting algorithm is applied to that hook) and not empty

```
1006      \bool_lazy_and:nnTF
1007          { \__hook_if_usable_p:n {#1} }
1008          { ! \hook_if_empty_p:n {#1} }
1009          {
1010              \__hook_log_line:x
1011              {
1012                  Execution~order
1013                  \bool_if:NTF \l__hook_tmpa_bool
1014                      { \__hook_if_reversed:nT {#1} { ~(after~reversal) } }
1015                      { ~ (after~
1016                          \__hook_if_reversed:nT {#1} { reversal~and~ }
1017                          applying~rules)
1018                      } :
1019                  }
1020                  #2 % \tl_show:n
1021                  {
1022                      \@spaces
1023                      \clist_if_empty:cTF { g__hook_#1_labels_clist }
1024                          { --- }
1025                          { \clist_use:cn {g__hook_#1_labels_clist} { ,~ } }
1026                  }
1027                  }
1028                  {
1029                      \__hook_log_line:x { Execution~order: }
1030                      #2
1031                      {
1032                          \@spaces Not~set~because~the~hook~ \__hook_if_usable:nTF {#1}
1033                          { code~pool~is~empty }
1034                          { is~\__hook_if_disabled:nTF {#1} {disabled} {undeclared} }
```

```

1035         }
1036     }
1037 }
1038 }
```

To display the code for next invocation only (i.e., from `\AddToHookNext` we have to remove the first two tokens at the front which are `\tl_gclear:N` and the token list to clear.

```

1039 \cs_new:Npn \__hook_log_next_code:n #1
1040   { \exp_args:No \tl_to_str:n { \use_none:nn #1 } }
```

`__hook_log_next_code:n`

(End definition for `\hook_show:n` and others. These functions are documented on page 193.)

This macro takes a `\langle hook` and an `\langle inline function` and loops through each pair of `\langle labels` in the `\langle hook`, and if there is a relation between this pair of `\langle labels`, the `\langle inline function` is executed with `#1 = \langle relation`, `#2 = \langle label1 | \langle label2`, and `#3 = \langle hook` (the latter may be the argument `#1` to `__hook_list_rules:nn`, or `??` if it is a default rule).

```

1041 \cs_new_protected:Npn \__hook_list_rules:nn #1 #2
1042   {
1043     \cs_set_protected:Npn \__hook_tmp:w ##1 ##2 ##3 {##2}
1044     \prop_map_inline:cn { g__hook_#1_code_prop }
1045     {
1046       \prop_map_inline:cn { g__hook_#1_code_prop }
1047       {
1048         \__hook_if_label_case:nnnn {##1} {####1}
1049         { \prop_map_break: }
1050         { \__hook_list_one_rule:nnn {##1} {####1} }
1051         { \__hook_list_one_rule:nnn {####1} {##1} }
1052         {##1}
1053       }
1054     }
1055   }
```

These two are quite similar to `__hook_apply_label_pair:nnn` and `__hook_label_if_exist_apply:nnnF`, respectively, but rather than applying the rule, they pass it to the `\langle inline function`.

```

1056 \cs_new_protected:Npn \__hook_list_one_rule:nnn #1#2#3
1057   {
1058     \__hook_list_if_rule_exists:nnnF {#1} {#2} {#3}
1059     { \__hook_list_if_rule_exists:nnnF {#1} {#2} { ?? } { } }
1060   }
1061 \cs_new_protected:Npn \__hook_list_if_rule_exists:nnnF #1#2#3
1062   {
1063     \if_cs_exist:w g__hook_ #3 _rule_ #1 | #2 _tl \cs_end:
1064     \exp_args:Nv \__hook_tmp:w
1065     { g__hook_ #3 _rule_ #1 | #2 _tl } { #1 | #2 } {#3}
1066     \exp_after:wN \use_none:nn
1067   \fi:
1068   \use:n
1069 }
```

(End definition for `__hook_list_rules:nn`, `__hook_list_one_rule:nnn`, and `__hook_list_if_rule_exists:nnnF`.)

```

\__hook_debug_print_rules:n A shorthand for debugging that prints similar to \prop_show:N.
1070 \cs_new_protected:Npn \__hook_debug_print_rules:n #1
1071 {
1072     \iow_term:n { The~hook~#1~contains~the~rules: }
1073     \cs_set_protected:Npn \__hook_tmp:w ##1
1074     {
1075         \__hook_list_rules:nn {#1}
1076         {
1077             \iow_term:x
1078             {
1079                 > ##1 {####2} ##1 => ##1 {####1}
1080                 \str_if_eq:nnT {####3} {??} { ~ (default) }
1081             }
1082         }
1083     }
1084     \exp_args:No \__hook_tmp:w { \use:nn { ~ } { ~ } }
1085 }

```

(End definition for `__hook_debug_print_rules:n`.)

4.8 Specifying code for next invocation

```

\hook_gput_next_code:nn
1086 \cs_new_protected:Npn \hook_gput_next_code:nn #1
1087     { \__hook_normalize_hook_args:Nn \__hook_gput_next_code:nn {#1} }

(End definition for \hook_gput_next_code:nn. This function is documented on page 192.)

```

```

\__hook_gput_next_code:nn
\__hook_gput_next_do:nn
\__hook_gput_next_do:Nnn
1088 \cs_new_protected:Npn \__hook_gput_next_code:nn #1 #2
1089 {
1090     \__hook_if_disabled:nTF {#1}
1091     { \msg_error:nnn { hooks } { hook-disabled } {#1} }
1092     {
1093         \__hook_if_structure_exist:nTF {#1}
1094         { \__hook_gput_next_do:nn {#1} {#2} }
1095         { \__hook_try_declarating_generic_next_hook:nn {#1} {#2} }
1096     }
1097 }
1098 \cs_new_protected:Npn \__hook_gput_next_do:nn #1
1099 {
1100     \exp_args:Nc \__hook_gput_next_do:Nnn
1101     { __hook_next~#1 } {#1}
1102 }

```

First check if the “next code” token list is empty: if so we need to add a `\tl_gclear:c` to clear it, so the code lasts for one usage only. The token list is cleared early so that nested usages don’t get lost. `\tl_gclear:c` is used instead of `\tl_gclear:N` in case the hook is used in an expansion-only context, so the token list doesn’t expand before `\tl_gclear:N`: that would make an infinite loop. Also in case the main code token list is empty, the hook code has to be updated to add the next execution token list.

```

1103 \cs_new_protected:Npn \__hook_gput_next_do:Nnn #1 #2
1104 {

```

```

1105      \tl_if_empty:cT { __hook~#2 }
1106      { __hook_update_hook_code:n {#2} }
1107      \tl_if_empty:NT #1
1108      { __hook_tl_gset:Nn #1 { __hook_clear_next:n {#2} } }
1109      __hook_tl_gput_right:Nn #1
1110  }

(End definition for __hook_gput_next_code:nn, __hook_gput_next_do:nn, and
 __hook_gput_next_do:Nnn.)

```

`\hook_gclear_next_code:n` Discard anything set up for next invocation of the hook.

```

__hook_clear_next:n
1111 \cs_new_protected:Npn \hook_gclear_next_code:n #1
1112 { __hook_normalize_hook_args:Nn __hook_clear_next:n {#1} }
1113 \cs_new_protected:Npn __hook_clear_next:n #1
1114 { \cs_gset_eq:cN { __hook_next~#1 } \c_empty_tl }

(End definition for \hook_gclear_next_code:n and __hook_clear_next:n. This function is
documented on page 192.)

```

4.9 Using the hook

`\hook_use:n` `\hook_use:n` as defined here is used in the preamble, where hooks aren't initialized by default. `\hook_use_initialized:n` is also defined, which is the non-\protected version for use within the document. Their definition is identical, except for the `\hook_preamble_hook:n` (which wouldn't hurt in the expandable version, but it would be an unnecessary extra expansion).

`\hook_use_initialized:n` holds the expandable definition while in the preamble. `\hook_preamble_hook:n` initializes the hook in the preamble, and is redefined to `\use_none:n` at `\begin{document}`.

Both versions do the same thing internally: they check that the hook exists as given, and if so they use it as quickly as possible.

At `\begin{document}`, all hooks are initialized, and any change in them causes an update, so `\hook_use:n` can be made expandable. This one is better not protected so that it can expand into nothing if containing no code. Also important in case of generic hooks that we do not generate a `\relax` as a side effect of checking for a csname. In contrast to the TeX low-level `\csname ... \endcsname` construct `\tl_if_exist:c` is careful to avoid this.

```

1115 <latexrelease>\IncludeInRelease{2021/11/15}{\hook_use:n}
1116 <latexrelease>                                {Standardise-generic-hook-names}
1117 \cs_new_protected:Npn \hook_use:n #1
1118 {
1119     \tl_if_exist:cT { __hook~#1 }
1120     {
1121         __hook_preamble_hook:n {#1}
1122         \cs:w __hook~#1 \cs_end:
1123     }
1124 }
1125 \cs_new:Npn __hook_use_initialized:n #1
1126 {
1127     \if_cs_exist:w __hook~#1 \cs_end:
1128         \cs:w __hook~#1 \exp_after:wN \cs_end:
1129     \fi:
1130 }

```

```

1131 \cs_new_protected:Npn \__hook_preamble_hook:n #1
1132   { \__hook_initialize_hook_code:n {#1} }
1133 \EndIncludeInRelease
1134 \IncludeInRelease{2020/10/01}{\hook_use:n}
1135 \Standardise-generic-hook-names
1136 \cs_new_protected:Npn \hook_use:n #1
1137 \{
1138   \tl_if_exist:cTF { __hook~#1 }
1139   {
1140     \__hook_preamble_hook:n {#1}
1141     \cs:w __hook~#1 \cs_end:
1142   }
1143   { \__hook_use:wn #1 / \s__hook_mark {#1} }
1144 \}
1145 \cs_new:Npn \__hook_use_initialized:n #1
1146 \{
1147   \if_cs_exist:w __hook~#1 \cs_end:
1148   \else:
1149     \__hook_use_undefined:w
1150   \fi:
1151   \cs:w __hook~#1 \__hook_use_end:
1152 \}
1153 \cs_new:Npn \__hook_use_undefined:w #1 #2 __hook~#3 \__hook_use_end:
1154 \{
1155   #1 % fi
1156   \__hook_use:wn #3 / \s__hook_mark {#3}
1157 \}
1158 \cs_new_protected:Npn \__hook_preamble_hook:n #1
1159 \{ \__hook_initialize_hook_code:n {#1} \}
1160 \cs_new_eq:NN \__hook_use_end: \cs_end:
1161 \EndIncludeInRelease

```

(End definition for `\hook_use:n` and others. This function is documented on page 192.)

`__hook_use:wn` does a quick check to test if the current hook is a file hook: those need a special treatment. If it is not, the hook does not exist. If it is, then `__hook_try_file_hook:n` is called, and checks that the current hook is a file-specific hook using `__hook_if_file_hook:wTF`. If it's not, then it's a generic `file/` hook and is used if it exist.

If it is a file-specific hook, it passes through the same normalization as during declaration, and then it is used if defined. `__hook_if_usable_use:n` checks if the hook exist, and calls `__hook_preamble_hook:n` if so, then uses the hook.

```

1162 \IncludeInRelease{2021/11/15}{\__hook_use:wn}
1163 \Standardise-generic-hook-names
1164 \EndIncludeInRelease
1165 \IncludeInRelease{2020/10/01}{\__hook_use:wn}
1166 \Standardise-generic-hook-names
1167 \cs_new:Npn \__hook_use:wn #1 / #2 \s__hook_mark #3
1168 \{
1169   \str_if_eq:nnTF {#1} { file }
1170   { \__hook_try_file_hook:n {#3} }
1171   { } % Hook doesn't exist
1172 \}

```

```

1173 <latexrelease>\cs_new_protected:Npn \__hook_try_file_hook:n #1
1174 <latexrelease>  {
1175   <latexrelease>    \__hook_if_file_hook:wTF #1 / \s__hook_mark
1176   <latexrelease>    {
1177     <latexrelease>      \exp_args:Ne \__hook_if_usable_use:n
1178     <latexrelease>      { \exp_args:Ne \__hook_file_hook_normalize:n {#1} }
1179   <latexrelease>    }
1180   <latexrelease>    { \__hook_if_usable_use:n {#1} } % file/ generic hook (e.g. file/before)
1181 <latexrelease>  }

1182 <latexrelease>\cs_new_protected:Npn \__hook_if_usable_use:n #1
1183 <latexrelease>  {
1184   <latexrelease>    \tl_if_exist:cT { __hook~#1 }
1185   <latexrelease>    {
1186     <latexrelease>      \__hook_preamble_hook:n {#1}
1187     <latexrelease>      \cs:w __hook~#1 \cs_end:
1188   <latexrelease>    }
1189 <latexrelease>  }

1190 <latexrelease>\EndIncludeInRelease

```

(End definition for `__hook_use:wn`, `__hook_try_file_hook:n`, and `__hook_if_usable_use:n`.)

\hook_use_once:n `__hook_use_once:n` For hooks that can and should be used only once we have a special use command that further inhibits the hook from getting more code added to it. This has the effect that any further code added to the hook is executed immediately rather than stored in the hook.

The code needs some gymnastics to prevent space trimming from the hook name, since `\hook_use:n` and `\hook_use_once:n` are documented to not trim spaces.

```

1191 \cs_new_protected:Npn \hook_use_once:n #1
1192  {
1193   \__hook_if_execute_immediately:nF {#1}
1194   { \__hook_normalize_hook_args:Nn \__hook_use_once:n { \use:n {#1} } }
1195  }
1196 \cs_new_protected:Npn \__hook_use_once:n #1
1197  {
1198   \__hook_preamble_hook:n {#1}
1199   \__hook_use_once_set:n {#1}
1200   \__hook_use_initialized:n {#1}
1201   \__hook_use_once_clear:n {#1}
1202  }

```

`__hook_use_once_set:n` is used before the actual hook code is executed so that any usage of `\AddToHook` inside the hook causes the code to execute immediately. Setting `\g__hook_<hook>_reversed_tl` to `I` prevents further code from being added to the hook. `__hook_use_once_clear:n` then clears the hook so that any further call to `\hook_use:n` or `\hook_use_once:n` will expand to nothing.

```

1203 \cs_new_protected:Npn \__hook_use_once_set:n #1
1204  { \__hook_tl_gset:cn { g__hook_#1_reversed_tl } { I } }
1205 \cs_new_protected:Npn \__hook_use_once_clear:n #1
1206  {
1207   \__hook_tl_gclear:c { __hook~#1 }
1208   \__hook_tl_gclear:c { __hook_next~#1 }
1209   \__hook_tl_gclear:c { __hook_toplevel~#1 }
1210   \prop_gclear:c { g__hook_#1_code_prop }

```

```
1211 }
```

(End definition for `\hook_use_once:n` and others. This function is documented on page 192.)

`_hook_if_execute_immediately:p:n`
`_hook_if_execute_immediately:nTF`

To check whether the code being added should be executed immediately (that is, if the hook is a one-time hook), we check if it's usable (it can't be one-time if it was not already usable), then we check that `\g__hook_<hook>_reversed_tl` is `I`. The gymnastics around `\if:w` is there to allow the `reversed` token list to be empty.

```
1212 \prg_new_conditional:Npnn \_hook_if_execute_immediately:n #1 { F, TF }
1213 {
1214     \_hook_if_usable:nTF {#1}
1215     {
1216         \exp_after:wN \_hook_clean_to_scan:w
1217         \if:w I \cs:w g__hook_#1_reversed_tl \cs_end:
1218             \s__hook_mark \prg_return_true:
1219         \else:
1220             \s__hook_mark \prg_return_false:
1221         \fi:
1222     }
1223     { \prg_return_false: }
1224 }
```

(End definition for `_hook_if_execute_immediately:nTF`.)

4.10 Querying a hook

Simpler data types, like token lists, have three possible states; they can exist and be empty, exist and be non-empty, and they may not exist, in which case emptiness doesn't apply (though `\tl_if_empty:N` returns false in this case).

Hooks are a bit more complicated: they have several other states as discussed in 4.4.2. A hook may exist or not, and either way it may or may not be empty (even a hook that doesn't exist may be non-empty) or may be disabled.

A hook is said to be empty when no code was added to it, either to its permanent code pool, or to its “next” token list. The hook doesn't need to be declared to have code added to its code pool (it may happen that a package *A* defines a hook `foo`, but it's loaded after package *B*, which adds some code to that hook. In this case it is important that the code added by package *B* is remembered until package *A* is loaded).

All other states can only be queried with internal tests as the different states are irrelevant for package code.

`\hook_if_empty_p:n` Test if a hook is empty (that is, no code was added to that hook). A `<hook>` being empty means that all three of its `\g__hook_<hook>_code_prop`, its `_hook_toplevel<hook>` and its `_hook_next<hook>` are empty.

```
1225 \prg_new_conditional:Npnn \hook_if_empty:n #1 { p , T , F , TF }
1226 {
1227     \_hook_if_structure_exist:nTF {#1}
1228     {
1229         \bool_lazy_and:nnTF
1230             { \prop_if_empty_p:c { g__hook_#1_code_prop } }
1231             {
1232                 \bool_lazy_and_p:nn
1233                     { \tl_if_empty_p:c { __hook_toplevel~#1 } }
1234                     { \tl_if_empty_p:c { __hook_next~#1 } }
```

```

1235         }
1236         { \prg_return_true: }
1237         { \prg_return_false: }
1238     }
1239     { \prg_return_true: }
1240 }
```

(End definition for `\hook_if_empty:nTF`. This function is documented on page 193.)

`_hook_if_usable_p:n` A hook is usable if the token list that stores the sorted code for that hook, `_hook <hook>`, exists. The property list `\g_hook_<hook>_code_prop` cannot be used here because often it is necessary to add code to a hook without knowing if such hook was already declared, or even if it will ever be (for example, in case the package that defines it isn't loaded).

```

1241 \prg_new_conditional:Npnn \_hook_if_usable:n #1 { p , T , F , TF }
1242 {
1243     \tl_if_exist:cTF { _hook~#1 }
1244     { \prg_return_true: }
1245     { \prg_return_false: }
1246 }
```

(End definition for `_hook_if_usable:nTF`.)

`_hook_if_structure_exist_p:n` An internal check if the hook has already its basic internal structure set up with `_hook_init_structure:n`. This means that the hook was already used somehow (a code chunk or rule was added to it), but it still wasn't declared with `\hook_new:n`.

```

1247 \prg_new_conditional:Npnn \_hook_if_structure_exist:n #1 { p , T , F , TF }
1248 {
1249     \prop_if_exist:cTF { g_hook_#1_code_prop }
1250     { \prg_return_true: }
1251     { \prg_return_false: }
1252 }
```

(End definition for `_hook_if_structure_exist:nTF`.)

`_hook_if_declared_p:n` Internal test to check if the hook was officially declared with `\hook_new:n` or a variant.

`_hook_if_declared:nTF`

```

1253 \prg_new_conditional:Npnn \_hook_if_declared:n #1 { p , T , F , TF }
1254 {
1255     \tl_if_exist:cTF { g_hook_#1_declared_tl }
1256     { \prg_return_true: }
1257     { \prg_return_false: }
1258 }
```

(End definition for `_hook_if_declared:nTF`.)

`_hook_if_reversed_p:n` An internal conditional that checks if a hook is reversed.

`_hook_if_reversed:nTF`

```

1259 \prg_new_conditional:Npnn \_hook_if_reversed:n #1 { p , T , F , TF }
1260 {
1261     \exp_after:wN \_hook_clean_to_scan:w
1262     \if:w - \cs:w g_hook_#1_reversed_tl \cs_end:
1263         \s_hook_mark \prg_return_true:
1264     \else:
1265         \s_hook_mark \prg_return_false:
1266     \fi:
1267 }
```

(End definition for `_hook_if_reversed:nTF`.)

`_hook_if_generic_p:n` An internal conditional that checks if a name belongs to a generic hook. The deprecated version needs to check if #3 is empty to avoid returning true on `file/before`, for example.

```
1268 \prg_new_conditional:Npnn \_hook_if_generic:n #1 { T, TF }
1269   { \_hook_if_generic:w #1 // \s_hook_mark }
1270 \cs_new:Npn \_hook_if_generic:w #1 / #2 / #3 / #4 \s_hook_mark
1271   {
1272     \cs_if_exist:cTF { c_hook_generic:#1/.#3_t1 }
1273       { \prg_return_true: }
1274       { \prg_return_false: }
1275   }
1276 \prg_new_conditional:Npnn \_hook_if_deprecated_generic:n #1 { T, TF }
1277   { \_hook_if_deprecated_generic:w #1 // \s_hook_mark }
1278 \cs_new:Npn \_hook_if_deprecated_generic:w #1 / #2 / #3 / #4 \s_hook_mark
1279   {
1280     \cs_if_exist:cTF { c_hook_deprecated:#1/.#2_t1 }
1281     {
1282       \tl_if_empty:nTF {#3}
1283         { \prg_return_false: }
1284         { \prg_return_true: }
1285     }
1286     { \prg_return_false: }
1287 }
```

(End definition for `_hook_if_generic:nTF` and `_hook_if_deprecated_generic:nTF`.)

`_hook_if_generic_reversed_p:n` An internal conditional that checks if a name belongs to a generic reversed hook.

```
1288 \prg_new_conditional:Npnn \_hook_if_generic_reversed:n #1 { T }
1289   { \_hook_if_generic_reversed:w #1 // \scan_stop: }
1290 \cs_new:Npn \_hook_if_generic_reversed:w #1 / #2 / #3 / #4 \scan_stop:
1291   {
1292     \if_charcode:w - \cs:w c_hook_generic:#1/.#3_t1 \cs_end:
1293       \prg_return_true:
1294     \else:
1295       \prg_return_false:
1296     \fi:
1297 }
```

(End definition for `_hook_if_generic_reversed:nTF`.)

4.11 Messages

Hook errors are LaTeX kernel errors:

```
1298 \prop_gput:Nnn \g_msg_module_type_prop { hooks } { LaTeX }
And so are kernel errors (this should move elsewhere eventually).
1299 \prop_gput:Nnn \g_msg_module_type_prop { latex2e } { LaTeX }
1300 \prop_gput:Nnn \g_msg_module_name_prop { latex2e } { kernel }
1301 \msg_new:nnnn { hooks } { labels-incompatible }
1302   {
1303     Labels~'#1'~and~'#2'~are~incompatible
1304     \str_if_eq:nnF {#3} {??} { ~in-hook~'#3' } .~
1305     \int_compare:nNnTF {#4} = { 1 }
```

```

1306      { The~ code~ for~ both~ labels~ will~ be~ dropped. }
1307      { You~ may~ see~ errors~ later. }
1308  }
1309  { LaTeX-found-two-incompatible-labels-in-the-same-hook.~
1310  This-indicates-an-incompatibility-between-packages. }
1311 \msg_new:nnn { hooks } { exists }
1312     { Hook~'#1'~ has~ already~ been~ declared. }
1313     { There~ already~ exists~ a~ hook~ declaration~ with~ this~
1314         name.\\
1315         Please~ use~ a~ different~ name~ for~ your~ hook.}
1316 \msg_new:nnn { hooks } { hook-disabled }
1317     { Cannot-add-code-to-disabled-hook~'#1'. }
1318  {
1319     The-hook~'#1'~you-tried-to-add-code-to-was-previously-disabled~
1320     with~\iow_char:N\hook_disable_generic:n~or~\iow_char:N\DisableGenericHook,~so~
1321     it~cannot~have~code~added~to~it.
1322  }
1323 \msg_new:nnn { hooks } { empty-label }
1324  {
1325     Empty~code~label~\msg_line_context:~-
1326     Using~'\_\_hook\_currname\_or\_default:'~instead.
1327  }
1328 \msg_new:nnn { hooks } { no-default-label }
1329  {
1330     Missing~(empty)~default-label~\msg_line_context:.~\\
1331     This~command~was~ignored.
1332  }
1333 \msg_new:nnn { hooks } { unknown-rule }
1334     { Unknown~ relationship~ '#3'~
1335         between~ labels~ '#2'~ and~ '#4'~
1336         \str_if_eq:nnF {#1} {??} { ~in~hook~'#1' }. ~
1337         Perhaps~ a~ misspelling?
1338  }
1339  {
1340     The~ relation~ used~ not~ known~ to~ the~ system.~ Allowed~ values~ are~
1341     'before'~ or~ '<',~
1342     'after'~ or~ '>',~
1343     'incompatible-warning',~
1344     'incompatible-error',~
1345     'voids'~ or~
1346     'unrelated'.
1347  }
1348 \msg_new:nnn { hooks } { misused-top-level }
1349  {
1350     Illegal~use~of~\iow_char:N \\AddToHook{#1}[top-level]{...}.\\
1351     'top-level'~is~reserved~for~the~user's~document.
1352  }
1353  {
1354     The~'top-level'~label~is~meant~for~user~code~only,~and~should~only~
1355     be~used~(sparingly)~in~the~main~document.~Use~the~default~label~
1356     '\_\_hook\_currname\_or\_default:'~for~this~\@cls@pkg,~or~another~
1357     suitable~label.

```

```

1358     }
1359 \msg_new:nnn { hooks } { set-top-level }
1360 {
1361   You~cannot~change~the~default~label~'#1'~'top-level'.~Illegal \\
1362   \use:nn { ~ } { ~ } \iow_char:N \\#2{#3} \\
1363   \msg_line_context:.
1364 }

1365 \msg_new:nnn { hooks } { extra-pop-label }
1366 {
1367   Extra~\iow_char:N \\PopDefaultHookLabel. \\
1368   This~command~will~be~ignored.
1369 }
1370 \msg_new:nnn { hooks } { missing-pop-label }
1371 {
1372   Missing~\iow_char:N \\PopDefaultHookLabel. \\
1373   The~label~'#1'~was~pushed~but~never~popped.~Something~is~wrong.
1374 }

1375 \msg_new:nnn { latex2e } { should-not-happen }
1376 {
1377   This~should~not~happen.~#1 \\
1378   Please~report~at~https://github.com/latex3/latex2e.
1379 }

1380 \msg_new:nnn { hooks } { activate-disabled }
1381 {
1382   Cannot~ activate~ hook~ '#1'~ because~ it~ is~ disabled!
1383 }

1384 \msg_new:nnn { hooks } { cannot-remove }
1385 {
1386   Cannot~remove~chunk~'#2'~from~hook~'#1'~because~
1387   \_hook_if_structure_exist:nTF {#1}
1388   { it~does~not~exist~in~that~hook. }
1389   { the~hook~does~not~exist. }
1390 }

1391 \msg_new:nnn { hooks } { generic-deprecated }
1392 {
1393   Generic~hook~'#1/#2/#3'~is~deprecated. \\
1394   Use~hook~'#1/#3/#2'~instead.
1395 }

```

4.12 L^AT_EX 2 _{ϵ} package interface commands

\NewHook

\NewReversedHook
\NewMirroredHookPair

Declaring new hooks ...

```

1396 \NewDocumentCommand \NewHook          { m }{ \hook_new:n {#1} }
1397 \NewDocumentCommand \NewReversedHook  { m }{ \hook_new_reversed:n {#1} }
1398 \NewDocumentCommand \NewMirroredHookPair { mm }{ \hook_new_pair:nn {#1}{#2} }

```

(End definition for \NewHook, \NewReversedHook, and \NewMirroredHookPair. These functions are documented on page 181.)

```

1399 \IfFileExists{ltexrelease}{\IncludeInRelease{2021/06/01}}%
1400 \IfFileExists{ltexrelease}{\hook_activate_generic:n}{Providing~hooks}

```

\ActivateGenericHook Providing new hooks ...

```
1401 \NewDocumentCommand \ActivateGenericHook { m }{ \hook_activate_generic:n {#1} }
```

(End definition for `\ActivateGenericHook`. This function is documented on page 182.)

\DisableGenericHook Disabling a generic hook.

```
1402 \NewDocumentCommand \DisableGenericHook { m }{ \hook_disable_generic:n {#1} }
```

(End definition for `\DisableGenericHook`. This function is documented on page 182.)

```
1403 <latexrelease>\EndIncludeInRelease
1404 <latexrelease>\IncludeInRelease{2020/10/01}
1405 <latexrelease>          {\hook_activate_generic:n}{Providing~hooks}
1406 <latexrelease>
1407 <latexrelease>\def \ActivateGenericHook#1{}
1408 <latexrelease>
1409 <latexrelease>\EndIncludeInRelease
```

\AddToHook

```
1410 \NewDocumentCommand \AddToHook { m o +m }
1411   { \hook_gput_code:nnn {#1} {#2} {#3} }
```

(End definition for `\AddToHook`. This function is documented on page 183.)

\AddToHookNext

```
1412 \NewDocumentCommand \AddToHookNext { m +m }
1413   { \hook_gput_next_code:nn {#1} {#2} }
```

(End definition for `\AddToHookNext`. This function is documented on page 185.)

\ClearHookNext

```
1414 \NewDocumentCommand \ClearHookNext { m }
1415   { \hook_gclear_next_code:n {#1} }
```

(End definition for `\ClearHookNext`. This function is documented on page 185.)

\RemoveFromHook

```
1416 \NewDocumentCommand \RemoveFromHook { m o }
1417   { \hook_gremove_code:nn {#1} {#2} }
```

(End definition for `\RemoveFromHook`. This function is documented on page 184.)

\SetDefaultHookLabel Now define a wrapper that replaces the top of the stack with the argument, and updates `\g__hook_hook_curr_name_tl` accordingly.

\PushDefaultHookLabel

\PopDefaultHookLabel

```
1418 \NewDocumentCommand \SetDefaultHookLabel { m }
1419   { \__hook_set_default_hook_label:n {#1} }
1420 %
1421 % The label is only automatically updated with \cs{@onefilewithoptions}
1422 % (\cs{usepackage} and \cs{documentclass}), but some packages, like
1423 % Ti\emph{k}Z, define package-like interfaces, like
1424 % \cs{usetikzlibrary} that are wrappers around \cs{input}, so they
1425 % inherit the default label currently in force (usually |top-level|,
1426 % but it may change if loaded in another package). To provide a
1427 % package-like behavior also for hooks in these files, we provide
1428 % high-level access to the default label stack.
1429 % \begin{macrocode}
```

```

1430 \NewDocumentCommand \PushDefaultHookLabel { m }
1431   { \__hook_curr_name_push:n {#1} }
1432 \NewDocumentCommand \PopDefaultHookLabel { }
1433   { \__hook_curr_name_pop: }

The current label stack holds the labels for all files but the current one (more or less like \currnamestack), and the current label token list, \g_hook_curr_name_tl, holds the label for the current file. However \pushfilename happens before \currname is set, so we need to look ahead to get the \currname for the label. expl3 also requires the current file in \pushfilename, so here we abuse \expl@push@filename@aux@@ to do \__hook_curr_name_push:n.

1434 \cs_gset_protected:Npn \expl@push@filename@aux@@ #1#2#3
1435   {
1436     \__hook_curr_name_push:n {#3}
1437     \str_gset:Nx \g_file_curr_name_str {#3}
1438     #1 #2 {#3}
1439   }

(End definition for \SetDefaultHookLabel, \PushDefaultHookLabel, and \PopDefaultHookLabel.  

These functions are documented on page 187.)

```

\UseHook Avoid the overhead of `xparse` and its protection that we don't want here (since the hook should vanish without trace if empty)!

```

1440 \cs_new:Npn \UseHook      { \hook_use:n }
1441 \cs_new:Npn \UseOneTimeHook { \hook_use_once:n }

```

(End definition for \UseHook and \UseOneTimeHook. These functions are documented on page 182.)

\ShowHook

```

1442 \cs_new_protected:Npn \ShowHook { \hook_show:n }
1443 \cs_new_protected:Npn \LogHook { \hook_log:n }

```

(End definition for \ShowHook and \LogHook. These functions are documented on page 190.)

\DebugHooksOn

\DebugHooksOff

```

1444 \cs_new_protected:Npn \DebugHooksOn { \hook_debug_on: }
1445 \cs_new_protected:Npn \DebugHooksOff { \hook_debug_off: }

```

(End definition for \DebugHooksOn and \DebugHooksOff. These functions are documented on page 191.)

\DeclareHookRule

```

1446 \NewDocumentCommand \DeclareHookRule { m m m m }
1447   { \hook_gset_rule:nnnn {#1}{#2}{#3}{#4} }

```

(End definition for \DeclareHookRule. This function is documented on page 188.)

\DeclareDefaultHookRule This declaration is only supported before `\begin{document}`.

```

1448 \NewDocumentCommand \DeclareDefaultHookRule { m m m }
1449   { \hook_gset_rule:nnnn {??}{#1}{#2}{#3} }
1450 \onlypreamble\DeclareDefaultHookRule

```

(End definition for \DeclareDefaultHookRule. This function is documented on page 189.)

\ClearHookRule A special setup rule that removes an existing relation. Basically @@_rule_gclear:nnn plus fixing the property list for debugging.

FMi: Needs perhaps an L3 interface, or maybe it should get dropped?

```
1451 \NewDocumentCommand \ClearHookRule { m m m }
1452 { \hook_gset_rule:nnn {#1}{#2}{unrelated}{#3} }
```

(End definition for `\ClearHookRule`. This function is documented on page 189.)

\IfHookEmptyTF Here we avoid the overhead of `xparse`, since `\IfHookEmptyTF` is used in `\end` (that is, every `LATEX` environment). As a further optimization, use `\let` rather than `\def` to avoid one expansion step.

```
1453 \cs_new_eq:NN \IfHookEmptyTF \hook_if_empty:nTF
```

(End definition for `\IfHookEmptyTF`. This function is documented on page 189.)

\IfHookExistsTF Marked for removal and no longer documented in the doc section!

PhO: \IfHookExistsTF is used in `jlreq.cls`, `pxatbegshi.sty`, `pxeverysel.sty`, `pxeveryshi.sty`, so the public name may be an alias of the internal conditional for a while. Regardless, those packages' use for \IfHookExistsTF is not really correct and can be changed.

```
1454 \cs_new_eq:NN \IfHookExistsTF \__hook_if_usable:nTF
```

(End definition for `\IfHookExistsTF`.)

4.13 Deprecated that needs cleanup at some point

`\hook_disable:n` Deprecated.

```
\hook_provide:n 1455 \cs_new_protected:Npn \hook_disable:n
\hook_provide_reversed:n 1456 {
  1457   \__hook_deprecated_warn:nn
  1458   { hook_disable:n }
  1459   { hook_disable_generic:n }
  1460   \hook_disable_generic:n
  1461 }
  1462 \cs_new_protected:Npn \hook_provide:n
  1463 {
    1464   \__hook_deprecated_warn:nn
    1465   { hook_provide:n }
    1466   { hook_activate_generic:n }
    1467   \hook_activate_generic:n
  1468 }
  1469 \cs_new_protected:Npn \hook_provide_reversed:n
  1470 {
    1471   \__hook_deprecated_warn:nn
    1472   { hook_provide_reversed:n }
    1473   { hook_activate_generic:n }
    1474   \__hook_activate_generic_reversed:n
  1475 }
  1476 \cs_new_protected:Npn \hook_provide_pair:nn
  1477 {
    1478   \__hook_deprecated_warn:nn
    1479   { hook_provide_pair:nn }
    1480   { hook_activate_generic:n }
    1481   \__hook_activate_generic_pair:nn
  1482 }
  1483 \cs_new_protected:Npn \__hook_activate_generic_reversed:n #1
```

```

1484 { \__hook_normalize_hook_args:Nn \__hook_activate_generic:nn {#1} { - } }
1485 \cs_new_protected:Npn \__hook_activate_generic_pair:nn #1#2
1486 { \hook_activate_generic:n {#1} \__hook_activate_generic_reversed:n {#2} }

(End definition for \hook_disable:n and others.)

\DisableHook Deprecated.
\ProvideHook
\ProvideReversedHook
\ProvideMirroredHookPair
1487 \cs_new_protected:Npn \DisableHook
1488 {
1489     \__hook_deprecated_warn:nn
1490     { DisableHook }
1491     { DisableGenericHook }
1492     \hook_disable_generic:n
1493 }
1494 \cs_new_protected:Npn \ProvideHook
1495 {
1496     \__hook_deprecated_warn:nn
1497     { ProvideHook }
1498     { ActivateGenericHook }
1499     \hook_activate_generic:n
1500 }
1501 \cs_new_protected:Npn \ProvideReversedHook
1502 {
1503     \__hook_deprecated_warn:nn
1504     { ProvideReversedHook }
1505     { ActivateGenericHook }
1506     \__hook_activate_generic_reversed:n
1507 }
1508 \cs_new_protected:Npn \ProvideMirroredHookPair
1509 {
1510     \__hook_deprecated_warn:nn
1511     { ProvideMirroredHookPair }
1512     { ActivateGenericHook }
1513     \__hook_activate_generic_pair:nn
1514 }

(End definition for \DisableHook and others.)

\__hook_DEPRECATED_WARN:NN Warns about a deprecation, telling what should be used instead.
1515 \cs_new_protected:Npn \__hook_DEPRECATED_WARN:NN #1 #2
1516 { \msg_warning:nnnn { hooks } { deprecated } {#1} {#2} }
1517 \msg_new:nnn { hooks } { deprecated }
1518 {
1519     Command~\iow_char:N\\#1~is~deprecated~and~will~be~removed~in~a~
1520     future~release. \\ \\
1521     Use~\iow_char:N\\#2~instead.
1522 }

(End definition for \__hook_DEPRECATED_WARN:NN.)

```

4.14 Internal commands needed elsewhere

Here we set up a few horrible (but consistent) L^AT_EX 2 _{ε} names to allow for internal commands to be used outside this module. We have to unset the `@@` since we want double “at” sign in place of double underscores.

```
1523 <@=
```

```
\@expl@@@initialize@all@@
```

```
1524 \cs_new_eq:NN \@expl@@@initialize@all@@  
1525         \__hook_initialize_all:  
1526 \cs_new_eq:NN \@expl@@@hook@curr@name@pop@@  
1527         \__hook_curr_name_pop:
```

(End definition for \@expl@@@initialize@all@@ and \@expl@@@hook@curr@name@pop@@.)

Rolling back here doesn't undefine the interface commands as they may be used in packages without rollback functionality. So we just make them do nothing which may or may not work depending on the code usage.

```
1528 %  
1529 <latexrelease>\IncludeInRelease{0000/00/00}%  
1530 <latexrelease>           {lthooks}{The~hook~management}%  
1531 <latexrelease>  
1532 <latexrelease>\def \NewHook#1{}  
1533 <latexrelease>\def \NewReversedHook#1{}  
1534 <latexrelease>\def \NewMirroredHookPair#1#2{}  
1535 <latexrelease>  
1536 <latexrelease>\def \DisableGenericHook #1{}  
1537 <latexrelease>  
1538 <latexrelease>\long\def\AddToHookNext#1#2{}  
1539 <latexrelease>  
1540 <latexrelease>\def\AddToHook#1{\gobble@AddToHook@args}  
1541 <latexrelease>\providecommand\gobble@AddToHook@args [2] {}  
1542 <latexrelease>  
1543 <latexrelease>\def\RemoveFromHook#1{\gobble@RemoveFromHook@arg}  
1544 <latexrelease>\providecommand\gobble@RemoveFromHook@arg[1] {}  
1545 <latexrelease>  
1546 <latexrelease>\def \UseHook      #1{}  
1547 <latexrelease>\def \UseOneTimeHook #1{}  
1548 <latexrelease>\def \ShowHook #1{}  
1549 <latexrelease>\let \DebugHooksOn \empty  
1550 <latexrelease>\let \DebugHooksOff \empty  
1551 <latexrelease>  
1552 <latexrelease>\def \DeclareHookRule #1#2#3#4{}  
1553 <latexrelease>\def \DeclareDefaultHookRule #1#2#3{}  
1554 <latexrelease>\def \ClearHookRule #1#2#3{}
```

If the hook management is not provided we make the test for existence false and the test for empty true in the hope that this is most of the time reasonable. If not a package would need to guard against running in an old kernel.

```
1555 <latexrelease>\long\def \IfHookExistsTF #1#2#3{#3}  
1556 <latexrelease>\long\def \IfHookEmptyTF #1#2#3{#2}  
1557 <latexrelease>  
1558 <latexrelease>\EndModuleRelease  
1559 \ExplSyntaxOff  
1560 </2ekernel | latexrelease>
```

File i

ltcmdhooks.dtx

Contents

1 Introduction

This file implements generic hooks for (arbitrary) commands. In theory every command `\langle name \rangle` offers now two associated hooks to which code can be added using `\AddToHook` or `\AddToHookNext`.¹¹ These are

`cmd/\langle name \rangle/before` This hook is executed at the very start of the command execution after its arguments (if any) are parsed. The hook `\langle code \rangle` is wrapped in the command inside a call to `\UseHook{cmd/\langle name \rangle/before}`, so the arguments passed to the command are *not* available in the hook `\langle code \rangle`.

`cmd/\langle name \rangle/after` This hook is similar to `cmd/\langle name \rangle/before`, but it is executed at the very end of the command body. This hook is implemented as a reversed hook.

The hooks are not physically present before `\begin{document}` (i.e., using a command in the preamble will never execute them) and if nobody has declared any code for them, then they are not added to the command code ever. For example, if we have the following definition

```
\newcommand\foo[2]{Code #1 for #2!}
```

then executing `\foo{A}{B}` will simply run `Code_A_for_B!` as it was always the case. However, if somebody, somewhere (e.g., in a package) adds

```
\AddToHook{cmd/foo/before}{<before code>}
```

then, after `\begin{document}` the definition of `\foo` will be:

```
\renewcommand\foo[2]{\UseHook{cmd/foo/before}Code #1 for #2!}
```

and similarly `\AddToHook{cmd/foo/after}{<after code>}` alters the definition to

```
\renewcommand\foo[2]{Code #1 for #2!\UseHook{cmd/foo/after}}
```

In other words, the mechanism is similar to what `etoolbox` offers with `\preto` and `\appto` with the important differences

- that code can be prepended or appended (i.e., added to the hooks) even if the command itself is not defined, because the defining package has not yet been loaded
- and that by using the hook management interface it is now possible to define how the code chunks added in these places are ordered, if different packages want to add code at these points.

¹¹In practice this is not supported for all types of commands, see section 2.2 for the restrictions that apply and what happens if one tries to use this with commands for which this is not supported.

2 Restrictions and Operational details

Adding arbitrary material to commands is tricky because most of the time we do not know what the macro expects as arguments when expanding and TeX doesn't have a reliable way to see that, so some guesswork has to be employed.

2.1 Patching

The code here tries to find out if a command was defined with `\newcommand` or `\DeclareRobustCommand` or `\NewDocumentCommand`, and if so it *assumes* that the argument specification of the command is as expected (which is not fail-proof, if someone redefines the internals of these commands in devious ways, but is a reasonable assumption).

If the command is one of the defined types, the code here does a sandboxed expansion of the command such that it can be redefined again exactly as before, but with the hook code added.

If however the command is not a known type (it was defined with `\def`, for example), then the code uses an approach similar to `etoolbox`'s `\patchcmd` to retokenize the command with the hook code in place. This procedure, however, is more likely to fail if the catcode settings are not the same as the ones at the time of command's definition, so not always adding a hook to a command will work.

2.1.1 Timing

When `\AddToHook` (or its `expl3` equivalent) is called with a generic `cmd` hook, say, `cmd/foo/before`, for the first time (that is, no code was added to that same hook before), in the preamble of a document, it will store a patch instruction for that command until `\begin{document}`, and only then all the commands which had hooks added will be patched in one go. That means that no command in the preamble will have hooks patched into them.

At `\begin{document}` all the delayed patches will be executed, and if the command doesn't exist the code is still added to the hook, but it will not be executed. After `\begin{document}`, when `\AddToHook` is called with a generic `cmd` hook the first time, the command will be immediately patched to include the hook, and if it doesn't exist or if it can't be patched for any reason, an error is thrown; if `\AddToHook` was already used in the preamble no new patching is attempted.

This has the consequence that a command defined or redefined after `\begin{document}` only uses generic `cmd` hook code if `\AddToHook` is called for the first time after the definition is made, or if the command explicitly uses the generic hook in its definition by declaring it with `\NewHookPair` adding `\UseHook` as part of the code.¹²

2.2 Commands that look ahead

Some commands are defined in different “steps” and they look ahead in the input stream to find more arguments. If you try to add some code to the `cmd/<name>/after` hook of such command, it will not work, and it is not possible to detect that programmatically, so the user has to know (or find out) which commands can or cannot have hooks attached to them.

¹²We might change this behavior in the main document slightly after gaining some usage experience.

One good example is the `\section` command. You can add something to the `cmd/section/before` hook, but if you try to add something to the `cmd/section/after` hook, `\section` will no longer work. That happens because the `\section` macro takes no argument, but instead calls a few internal L^AT_EX macros to look for the optional and mandatory arguments. By adding code to the `cmd/section/after` hook, you get in the way of that scanning.

3 Package Author Interface

The `cmd` hooks are, by default, available for all commands that can be patched to add the hooks. For some commands, however, the very beginning or the very end of the code is not the best place to put the hooks, for example, if the command looks ahead for arguments (see section 2.2).

If you are a package author and you want to add the hooks to your own commands in the proper position you can define the command and manually add the `\UseHook` calls inside the command in the proper positions, and manually define the hooks with `\NewHook` or `\NewReversedHook`. When the hooks are explicitly defined, patching is not attempted so you can make sure your command works properly. For example, an (admittedly not really useful) command that typesets its contents in a framed box with width optionally given in parentheses:

```
\newcommand\fancybox{\@ifnextchar({\@fancybox}{\@fancybox(5cm)}}
\def\@fancybox(#1)#2{\fbox{\parbox{#1}{#2}}}
```

If you try that definition, then add some code after it with

```
\AddToHook{cmd/fancybox/after}{<code>}
```

and then use the `\fancybox` command you will see that it will be completely broken, because the hook will get executed in the middle of parsing for optional (...) argument.

If, on the other hand, you want to add hooks to your command you can do something like:

```
\newcommand\fancybox{\@ifnextchar({\@fancybox}{\@fancybox(5cm)}}
\def\@fancybox(#1)#2{\fbox{%
    \UseHook{cmd/fancybox/before}%
    \parbox{#1}{#2}%
    \UseHook{cmd/fancybox/after}}}
\NewHook{cmd/fancybox/before}
\NewReversedHook{cmd/fancybox/after}
```

then the hooks will be executed where they should and no patching will be attempted. It is important that the hooks are declared with `\NewHook` or `\NewReversedHook`, otherwise the command hook code will try to patch the command. Note also that the call to `\UseHook{cmd/fancybox/before}` does not need to be in the definition of `\fancybox`, but anywhere it makes sense to insert it (in this case in the internal `\@fancybox`).

Alternatively, if for whatever reason your command does not support the generic hooks provided here, you can disable a hook with `\DisableHook`¹³, so that when someone

¹³Please use `\DisableHook` if at all, only on hooks that you “own”, i.e., for commands that your package or class defines and not second guess whether or not hooks of other packages should get disabled!

tries to add code to it they will get an error. Or if you don't want the error, you can simply declare the hook with `\NewHook` and never use it.

The above approach is useful for really complex commands where for one or the other reason the hooks can't be placed at the very beginning and end of the command body and some hand-crafting is needed. However, in the example above the real (and in fact only) issue is the cascading argument parsing in the style developed long ago in L^AT_EX 2.09. Thus, a much simpler solution for this case is to replace it with the modern `\NewDocumentCommand` syntax and define the command as follows:

```
\DeclareDocumentCommand\fancybox{D(){5cm}m}{\fbox{\parbox{#1}{#2}}}
```

If you do that then both hooks automatically work and are patched into the right places.

4 The Implementation

4.1 Execution plan

To add `before` and `after` hooks to a command we will need to peek into the definition of a command, which is always a tricky thing to do. Some cases are easy because we know how the command was defined, so we can assume how its *(parameter text)* looks like (for example a command defined with `\newcommand` may have an optional argument followed by a run of mandatory arguments), so we can just expand that command and make it grab #1, #2, etc. as arguments and define it all back with the hooks added.

Life's usually not that easy, so with some commands we can't do that (a #1 might as well be #₁₂1₁₂ instead of the expected #₆1₁₂, for example) so we need to resort to "patching" the command: read its `\meaning`, and tokenize it again with `\scantokens` and hope for the best.

So the overall plan is:

1. Check if a command is of a known type (that is, defined with `\newcommand`¹⁴, `\DeclareRobustCommand`, or `\New(Expandable)DocumentCommand`), and if is, take appropriate action.
2. If the command is not a known type, we'll check if the command can be patched. Two things will prevent a command from being patched: if it was defined in a nonstandard catcode setting, or if it is an internal expl3 command with `--(module)` in its name, in which case we refuse to patch.
3. If the command was defined in nonstandard catcode settings, we will try a few standard ones to try our best to carry out the pathing. If this doesn't help either, the code will give up and throw an error.

¹ `(@@=hook)`

² `(*2ekernel | latexrelease)`
³ `\ExplSyntaxOn`
⁴ `\textrm{(\textit{latexrelease}) \textit{NewModuleRelease}\{2021/06/01\}\{ltcmdhooks}}`
⁵ `\textrm{(\textit{latexrelease}) \textit{The~hook~management~system~for~commands}}`

¹⁴It's not always possible to reliably detect this case because a command defined with no optional argument is indistinguishable from a `\def`ed command.

4.2 Variables

\g_hook_patch_action_list_tl	Pairs of \if<cmd>.. \patch<cmd> to be used with \robust@command@act when looking for a known patching rule. This token list is exposed because we see some future applications (with very specialized packages, such as etoolbox that may want to extend the pairs processed. It is not meant for general use which is why it is not documented in the interface documentation above.
	6 \tl_new:N \g_hook_patch_action_list_tl (End definition for \g_hook_patch_action_list_tl.)
\l__hook_patch_num_args_int	The number of arguments in a macro being patched. 7 \int_new:N \l__hook_patch_num_args_int (End definition for \l__hook_patch_num_args_int.)
\l__hook_patch_prefixes_tl \l__hook_param_text_tl \l__hook_replace_text_tl	The prefixes and parameters of the definition for the macro being patched. 8 \tl_new:N \l__hook_patch_prefixes_tl 9 \tl_new:N \l__hook_param_text_tl 10 \tl_new:N \l__hook_replace_text_tl (End definition for \l__hook_patch_prefixes_tl, \l__hook_param_text_tl, and \l__hook_replace_text_tl.)
\c__hook_hash_tl	A constant token list that contains two parameter tokens. 11 \tl_const:Nn \c__hook_hash_tl { # # } (End definition for \c__hook_hash_tl.)
__hook_exp_not:NN __hook_def_cmd:w	Two temporary macros that change depending on the macro being patched. 12 \cs_new_eq:NN __hook_exp_not:NN ? 13 \cs_new_eq:NN __hook_def_cmd:w ? (End definition for __hook_exp_not:NN and __hook_def_cmd:w.)
\q__hook_recursion_tail \q__hook_recursion_stop	Internal quarks for recursion: they can't appear in any macro being patched. 14 \quark_new:N \q__hook_recursion_tail 15 \quark_new:N \q__hook_recursion_stop (End definition for \q__hook_recursion_tail and \q__hook_recursion_stop.)
\g__hook_delayed_patches_prop	A list containing the patches delayed to \begin{document}, so that patching is not attempted twice. 16 \prop_new:N \g__hook_delayed_patches_prop (End definition for \g__hook_delayed_patches_prop.)
__hook_patch_debug:x	A helper for patching debug info. 17 \cs_new_protected:Npn __hook_patch_debug:x #1 18 { __hook_debug:n { \iow_term:x { [lthooks]~#1 } } } (End definition for __hook_patch_debug:x.)

4.3 Variants

```
\tl_rescan:nV \expl3 function variants used throughout the code.
19  \cs_generate_variant:Nn \tl_rescan:nn { nV }

(End definition for \tl_rescan:nV.)
```

4.4 Patching or delaying

Before `\begin{document}` all patching is delayed.

```
\_hook_try_put_cmd_hook:n  
\_hook_try_put_cmd_hook:w
```

This function is called from within `\AddToHook`, when code is first added to a generic cmd hook. If it is called within in the preamble, it delays the action until `\begin{document}`; otherwise it tries to update the hook.

```
20 <texrelease>\IncludeInRelease{2021/11/15}{\_hook_try_put_cmd_hook:n}%
21 <texrelease>          {Standardise-generic-hook-names}
22 \cs_new_protected:Npn \_hook_try_put_cmd_hook:n #1
23   { \_hook_try_put_cmd_hook:w #1 // \s_hook_mark {#1} }
24 \cs_new_protected:Npn \_hook_try_put_cmd_hook:w
25   #1 / #2 / #3 / #4 \s_hook_mark #5
26   {
27     \_hook_debug:n { \iow_term:n { ->~Adding~cmd~hook~to~'#2'~(#3): } }
28     \exp_args:Nc \_hook_patch_cmd_or_delay:Nnn {#2} {#2} {#3}
29   }
30 <texrelease>\EndIncludeInRelease
31 <texrelease>\IncludeInRelease{2021/06/01}{\_hook_try_put_cmd_hook:n}%
32 <texrelease>          {Standardise-generic-hook-names}
33 \cs_new_protected:Npn \_hook_try_put_cmd_hook:n #1
34 <texrelease>   { \_hook_try_put_cmd_hook:w #1 // \s_hook_mark {#1} }
35 \cs_new_protected:Npn \_hook_try_put_cmd_hook:w
36   #1 / #2 / #3 / #4 \s_hook_mark #5
37 <texrelease>   {
38   \_hook_debug:n { \iow_term:n { ->~Adding~cmd~hook~to~'#2'~(#3): } }
39   \str_case:nnTF {#3}
40     { { before } { } { after } { } }
41     { \exp_args:Nc \_hook_patch_cmd_or_delay:Nnn {#2} {#2} {#3} }
42     { \msg_error:nnnn { hooks } { wrong-cmd-hook } {#2} {#3} }
43   }
44 <texrelease>\EndIncludeInRelease

(End definition for \_hook_try_put_cmd_hook:n and \_hook_try_put_cmd_hook:w.)
```

```
\_hook_patch_cmd_or_delay:Nnn  
\_hook_cmd_begin_document_code:
```

In the preamble, `_hook_patch_cmd_or_delay:Nnn` just adds the patch instruction to a property list to be executed later.

```
45 \cs_new_protected:Npn \_hook_patch_cmd_or_delay:Nnn #1 #2 #3
46   {
47     \_hook_debug:n { \iow_term:n { ->~Add~generic~cmd~hook~for~#2~(#3). } }
48     \_hook_debug:n
49       { \iow_term:n { !~In~the~preamble:~delaying. } }
50     \prop_gput:Nnn \g_hook_delayed_patches_prop { #2 / #3 }
51       { \_hook_cmd_try_patch:nn {#2} {#3} }
52   }
```

The delayed patches are added to a property list to prevent duplication, and the code stored in the property list for each key is executed. The function `__hook_patch_cmd_or_delay:Nnn` is also redefined to be `__hook_patch_command:Nnn` so that no further delaying is attempted.

```

53 \cs_new_protected:Npn \_\_hook_cmd_begindocument_code:
54 {
55   \cs_gset_eq:NN \_\_hook_patch_cmd_or_delay:Nnn \_\_hook_patch_command:Nnn
56   \prop_map_function:NN \g_\_\_hook_delayed_patches_prop { \use_i:nn }
57   \prop_gclear:N \g_\_\_hook_delayed_patches_prop
58   \cs_undefine:N \_\_hook_cmd_begindocument_code:
59 }
60 \g@addto@macro \g@kernel@after@begindocument
61 { \_\_hook_cmd_begindocument_code: }

(End definition for \_\_hook_patch_cmd_or_delay:Nnn and \_\_hook_cmd_begindocument_code..)

```

`__hook_cmd_try_patch:nn` At `\begin{document}` tries patching the command if the hook was not manually created in the meantime. If the document does not exist, no error is raised here as it may hook into a package that wasn't loaded. Hooks added to commands in the document body still raise an error if the command is not defined.

```

62 \cs_new_protected:Npn \_\_hook_cmd_try_patch:nn #1 #
63 {
64   \_\_hook_debug:n
65   { \iow_term:x { ->~\string\begin{document}~try~cmd / #1 / #2. } }
66   \_\_hook_if_declared:nTF { cmd / #1 / #2 }
67   {
68     \_\_hook_debug:n
69     { \iow_term:n { .->~Giving~up:~hook~already~created. } }
70   }
71   {
72     \cs_if_exist:cT {#1}
73     { \exp_args:Nc \_\_hook_patch_command:Nnn {#1} {#1} {#2} }
74   }
75 }

(End definition for \_\_hook_cmd_try_patch:nn.)

```

4.5 Patching commands

`__hook_patch_command:Nnn`
`__hook_patch_check>NNnn`
`__hook_if_public_command:NTF`
`__hook_if_public_command:w`

`__hook_patch_command:Nnn` will do some sanity checks on the argument to detect if it is possible to add hooks to the command, and raises an error otherwise. If the command can contain hooks, then it uses `\robust@command@act` to find out what type is the command, and patch it accordingly.

```

76 \cs_new_protected:Npn \_\_hook_patch_command:Nnn #1 #2 #
77 {
78   \_\_hook_patch_debug:x { analyzing~'\token_to_str:N #1' }
79   \_\_hook_patch_debug:x { \token_to_str:N #1 = \token_to_meaning:N #1 }
80   \_\_hook_patch_check>NNnn \cs_if_exist:NTF #1 { undef }
81   {
82     \_\_hook_patch_debug:x { ++control~sequence~is~defined }
83     \_\_hook_patch_check>NNnn \token_if_macro:NTF #1 { macro }
84     {
85       \_\_hook_patch_debug:x { ++control~sequence~is~a~macro }

```

```

86          \_\_hook_patch_check:NNnn \_\_hook_if_public_command:NTF #1 { expl3 }
87          {
88              \_\_hook_patch_debug:x { +--+macro~is~not~private }
89              \robust@command@act
90                  \g_hook_patch_action_list_tl #1
91                  \_\_hook_retokenize_patch:Nnn { #1 {#2} {#3} }
92          }
93      }
94  }
95 }
```

And here's the auxiliary used above:

```

96 \cs_new_protected:Npn \_\_hook_patch_check:NNnn #1 #2 #3 #4
97  {
98      #1 #2 {#4}
99      {
100          \msg_error:nnxx { hooks } { cant-patch }
101          { \token_to_str:N #2 } {#3}
102      }
103  }
```

and a conditional __hook_if_public_command:N to check if a command has __ in its name (no other checking is performed). Primitives with :D in their name could be included here, but they are already discarded in the \token_if_macro:NTF test above.

```

104 \use:x
105  {
106      \prg_new_protected_conditional:Npnn
107          \exp_not:N \_\_hook_if_public_command:N ##1 { TF }
108      {
109          \exp_not:N \exp_last_unbraced:Nf
110          \exp_not:N \_\_hook_if_public_command:w
111          { \exp_not:N \cs_to_str:N ##1
112          \tl_to_str:n { _ _ } \s__hook_mark
113      }
114  }
115 \exp_last_unbraced:NNNNo
116 \cs_new_protected:Npn \_\_hook_if_public_command:w
117     #1 \tl_to_str:n { _ _ } #2 \s__hook_mark
118  {
119      \tl_if_empty:nTF {#2}
120      { \prg_return_true: }
121      { \prg_return_false: }
122  }
```

(End definition for __hook_patch_command:Nnn and others.)

4.5.1 Patching by expansion and redefinition

\g_hook_patch_action_list_tl This is the list of known command types and the function that patches the command hooks into them. The conditionals are taken from \ShowCommand, \NewCommandCopy and __kernel_cmd_if_xparse:NTF defined in ltcmd.

```

123 \tl_gset:Nn \g_hook_patch_action_list_tl
124  {
125      { \c@if@DeclareRobustCommand \_\_hook_patch_DeclareRobustCommand:Nnn }
126      { \c@if@newcommand \_\_hook_patch_newcommand:Nnn }
```

```

127      { \__kernel_cmd_if_xparse:NTF \__hook_cmd_patch_xparse:Nnn }
128  }

```

(End definition for `\g_hook_patch_action_list_tl.`)

`__hook_patch_DeclareRobustCommand:Nnn` At this point we know that the commands can be patched by expanding then redefining. These are the cases of commands defined with `\newcommand` with an optional argument or with `\DeclareRobustCommand`.

With `__hook_patch_DeclareRobustCommand:Nnn` we check if the command has an optional argument (with a test counter-intuitively called `\@if@newcommand`; also make sure the command doesn't take args by calling `\robust@command@chk@safe`). If so, we pass the patching action to `__hook_patch_newcommand:Nnn`, otherwise we call the patching engine `__hook_patch_expand_redefine:NNnn \c_false_bool` to indicate that there is no optional argument.

```

129 \cs_new_protected:Npn \__hook_patch_DeclareRobustCommand:Nnn #1
130  {
131      \exp_args:Nc \__hook_patch_DeclareRobustCommand_aux:Nnn
132      { \cs_to_str:N #1 ~ }
133  }
134 \cs_new_protected:Npn \__hook_patch_DeclareRobustCommand_aux:Nnn #1
135  {
136      \robust@command@chk@safe #1
137      { \@if@newcommand #1 }
138      { \use_ii:nn }
139      { \__hook_patch_newcommand:Nnn }
140      { \__hook_patch_expand_redefine:NNnn \c_false_bool }
141      #1
142  }

```

(End definition for `__hook_patch_DeclareRobustCommand:Nnn.`)

`__hook_patch_newcommand:Nnn` If the command was defined with `\newcommand` and an optional argument, call the patching engine with a `\c_true_bool` to flag the presence of an optional argument, and with `\\command` to patch the actual code for `\command`.

```

143 \cs_new_protected:Npn \__hook_patch_newcommand:Nnn #1
144  {
145      \exp_args:NNc \__hook_patch_expand_redefine:NNnn \c_true_bool
146      { \c_backslash_str \cs_to_str:N #1 }
147  }

```

(End definition for `__hook_patch_newcommand:Nnn.`)

`__hook_cmd_patch_xparse:Nnn` And for commands defined by the `xparse` commands use this for patching:

```

148 \cs_new_protected:Npn \__hook_cmd_patch_xparse:Nnn #1
149  {
150      \exp_args:NNc \__hook_patch_expand_redefine:NNnn \c_false_bool
151      { \cs_to_str:N #1 ~ code }
152  }

```

(End definition for `__hook_cmd_patch_xparse:Nnn.`)

```
\_\_hook_patch_expand_redefine:Nnnn
\_\_hook_redefine_with_hooks:Nnnn
\_\_hook_make_prefixes:w
```

Now the real action begins. Here we have in #1 a boolean indicating if the command has a leading [...] -delimited argument, in #2 the command control sequence, in #3 the name of the command (note that #1 $\neq \backslash\text{csname}\#2\backslash\text{endcsname}$ at this point!), and in #4 the hook position, either `before` or `after`.

Patching with expansion+redefinition is trickier than it looks like at first glance. Suppose the simple definition:

```
\def\foo#1{#1##2}
```

When defined, its *<replacement text>* will be a token list containing:

```
out_param 1, mac_param #, character 2
```

Then, after expanding `\foo{##1}` (here `##` denotes a single #₆) we end up with a token list with `out_param 1` replaced:

```
mac_param #, character 1, mac_param #, character 2
```

that is, the definition would be:

```
\def\foo#1{#1#2}
```

which obviously fails, because the original input in the definition was `##` but T_EX reduced that to a single parameter token #₆ when carrying out the definition. That leaves no room for a clever solution with (say) `\unexpanded`, because anything that would double the second #₆, would also (incorrectly) double the first, so there's not much to do other than a manual solution.

There are three cases we can distinguish to make things hopefully faster on simpler cases:

1. a macro with no parameters;
2. a macro with no parameter tokens in its definition;
3. a macro with parameters *and* parameter tokens.

The first case is trivial: if the macro has no parameters, we can just use `\unexpanded` around it, and if there is a parameter token in it, it is handled correctly (the macro can be treated as a `t1` variable).

The second case requires looking at the *<replacement text>* of the macro to see if it has a parameter token in there. If it does not, then there is no worry, and the macro can be redefined normally (without `\unexpanded`).

The third case, as usual, is the devious one. Here we'll have to loop through the definition token by token, and double every parameter token, so that this case can be handled like the previous one.

```
153 \cs_new_protected:Npn \_\_hook_patch_expand_redefine:Nnnn #1 #2 #3 #4
154 {
155     \_\_hook_patch_debug:x { +--+command~can~be~patched~without~rescanning }
```

We'll start by counting the number of arguments in the command by counting the number of characters in the `\cs_argument_spec:N` of the macro, divided by two, and subtracting one if the command has an optional argument (that is, an extra [] in its *<parameter text>*).

```
156     \int_set:Nn \l_\_hook_patch_num_args_int
157     {
158         \exp_args:Nf \str_count:n { \cs_argument_spec:N #2 } / 2
159         \bool_if:NT #1 { -1 }
160     }
```

Now build two token lists:

\l__hook_param_text_t1 will contain the *<parameter text>* to be used when redefining the macro. It should be identical to the *<parameter text>* used when originally defining that macro.

\l__hook_replace_text_t1 will contain braced pairs of \c__hook_hash_t1*<num>* to feed to the macro when expanded. This token list as well as the previous will have the first item surrounded by [...] in the case of an optional argument.

The use of \c__hook_hash_t1 here is to differentiate actual parameters in the macro from parameter tokens in the original definition of the macro. Later on, \c__hook_hash_t1 is either replaced by actual parameter tokens, or expanded into them.

```
161     \int_compare:nNnTF { \l__hook_patch_num_args_int } > { \c_zero_int }
162     {
```

We'll first check if the command has any parameter token in its definition (feeding it empty arguments), and set \l__hook_exp_not:n accordingly. \l__hook_exp_not:n will be used later to either leave \c__hook_hash_t1 or expand it, and also to remember the result of \l__hook_if_has_hash:nTF to avoid testing twice (the test can be rather slow).

```
163     \tl_set:Nx \l__hook_tmpa_t1 { \bool_if:NTF #1 { [ ] } { { } } }
164     \int_step_inline:nnn { 2 } { \l__hook_patch_num_args_int }
165     { \tl_put_right:Nn \l__hook_tmpa_t1 { { } } }
166     \exp_args:NNo \exp_args:No \l__hook_if_has_hash:nTF
167     { \exp_after:wN #2 \l__hook_tmpa_t1 }
168     { \cs_set_eq:NN \l__hook_exp_not:n \exp_not:n }
169     { \cs_set_eq:NN \l__hook_exp_not:n \use:n }
170     \cs_set_protected:Npn \l__hook_tmp:w ##1 ##2
171     {
172     ##1 \l__hook_param_text_t1 { \use:n ##2 }
173     ##1 \l__hook_replace_text_t1 { \l__hook_exp_not:n {##2} }
174 }
```

Here we'll conditionally add [...] around the first parameter:

```
175     \bool_if:NTF #1
176     { \l__hook_tmp:w \tl_set:Nx { [ \c__hook_hash_t1 1 ] } }
177     { \l__hook_tmp:w \tl_set:Nx { { \c__hook_hash_t1 1 } } }
```

Then, for every parameter from the second, just add it normally:

```
178     \int_step_inline:nnn { 2 } { \l__hook_patch_num_args_int }
179     { \l__hook_tmp:w \tl_put_right:Nx { { \c__hook_hash_t1 ##1 } } }
```

Now, if the command has any parameter token in its definition (then \l__hook_exp_not:n is \exp_not:n), call \l__hook_double_hashes:n to double them, and replace every \c__hook_hash_t1 by #:

```
180     \tl_set:Nx \l__hook_replace_text_t1
181     { \exp_not:N #2 \exp_not:V \l__hook_replace_text_t1 }
182     \tl_set:Nx \l__hook_replace_text_t1
183     {
184     \token_if_eq_meaning:NNTF \l__hook_exp_not:n \exp_not:n
185     { \exp_args:NNV \exp_args:No \l__hook_double_hashes:n }
186     { \exp_args:NV \exp_not:o }
187     \l__hook_replace_text_t1
188 }
```

And now, set a few auxiliaries for the case that the macro has parameters, so it won't be passed through `\unexpanded` (twice):

```

189      \cs_set_eq:NN \__hook_def_cmd:w \tex_gdef:D
190      \cs_set_eq:NN \__hook_exp_not:NN \prg_do_nothing:
191  }
192 {

```

In the case the macro has no parameters, we'll treat it as a token list and things are much simpler (expansion control looks a bit complicated, but it's just a pair of `\exp_not:N` preventing another `\exp_not:n` from expanding):

```

193      \tl_clear:N \l__hook_param_text_tl
194      \tl_set_eq:NN \l__hook_replace_text_tl #2
195      \cs_set_eq:NN \__hook_def_cmd:w \tex_xdef:D
196      \cs_set:Npn \__hook_exp_not:NN ##1 { \exp_not:N ##1 \exp_not:N }
197  }

```

Before redefining, we need to also get the prefixes used when defining the command. Here we ensure that the `\escapechar` is printable, otherwise a macro defined with prefixes `\protected \long` will have it `\meaning` printed as `protectedlong`, making life unnecessarily complicated. Here the `\escapechar` is changed to `/`, then we loop between pairs of `/.../` extracting the prefixes.

```

198 \group_begin:
199   \int_set:Nn \tex_escapechar:D { '/' }
200   \use:x
201   {
202     \group_end:
203     \tl_set:Nx \exp_not:N \l__hook_patch_prefixes_tl
204     { \exp_not:N \__hook_make_prefixes:w \cs_prefix_spec:N #2 / / }
205   }

```

Finally, call `__hook_redefine_with_hooks:Nnnn` with the macro being redefined in #1, then `\UseHook{cmd/<name>/before}` in #2 or `\UseHook{cmd/<name>/after}` in #3 (one is always empty), and in #4 the `<replacement text>` of the macro.

```

206 \use:x
207 {
208   \__hook_redefine_with_hooks:Nnnn \exp_not:N #2
209   \str_if_eq:nnTF {#4} { after }
210   { \use_i:nn }
211   { \use:nn }
212   { { \__hook_exp_not:NN \exp_not:N \UseHook { cmd / #3 / #4 } } }
213   { { } }
214   { \__hook_exp_not:NN \exp_not:V \l__hook_replace_text_tl }
215 }
216 }

```

Now that all the needed tools are ready, without further ado we'll redefine the command. The definition uses the prefixes gathered in `\l__hook_patch_prefixes_tl`, a primitive `__hook_def_cmd:w` (which is `\tex_gdef:D` or `\tex_xdef:D`) to avoid adding extra prefixes, and the `<parameter text>` from `\l__hook_param_text_tl`.

Then finally, in the body of the definition, we insert #2, which is `cmd/#1/before` or empty, #4 which is the `<replacement text>`, and #3 which is `cmd/#1/after` or empty.

```

217 \cs_new_protected:Npn \__hook_redefine_with_hooks:Nnnn #1 #2 #3 #4
218   {
219     \l__hook_patch_prefixes_tl

```

```

220      \exp_after:wN \__hook_def_cmd:w
221      \exp_after:wN #1 \l__hook_param_text_tl
222      { #2 #4 #3 }
223  }

```

Here's the auxiliary that makes the prefix control sequences for the redefinition. Each item has to be `\tl_trim_spaces:n`'d because the last item (and not any other) has a trailing space.

```

224 \cs_new:Npn \__hook_make_prefixes:w / #1 /
225 {
226   \tl_if_empty:nF {#1}
227   {
228     \exp_not:c { \tex_ \tl_trim_spaces:n {#1} :D }
229     \__hook_make_prefixes:w /
230   }
231 }

```

(End definition for `__hook_patch_expand_redefine:Nnnn`, `__hook_redefine_with_hooks:Nnnn`, and `__hook_make_prefixes:w`)

Here are some auxiliaries for the contraption above.

`__hook_if_has_hash_p:n` `__hook_if_has_hash:nTF` searches the token list #1 for a catcode 6 token, and if any is found, it returns `true`, and `false` otherwise. The searching doesn't care about preserving groups or spaces: we can ignore those safely (braces are removed) so that searching is as fast as possible.

```

232 \prg_new_conditional:Npnn \__hook_if_has_hash:n #1 { TF }
233   { \__hook_if_has_hash:w #1 ## \s__hook_mark }
234 \cs_new:Npn \__hook_if_has_hash:w #1
235   {
236     \tl_if_single_token:nTF {#1}
237     {
238       \token_if_eq_catcode:NNTF ## #1
239       { \__hook_if_has_hash_check:w }
240       { \__hook_if_has_hash:w }
241     }
242     { \__hook_if_has_hash:w #1 }
243   }
244 \cs_new:Npn \__hook_if_has_hash_check:w #1 \s__hook_mark
245   { \tl_if_empty:nTF {#1} { \prg_return_false: } { \prg_return_true: } }

```

(End definition for `__hook_if_has_hash:nTF`, `__hook_if_has_hash:w`, and `__hook_if_has_hash_check:w`)

`__hook_double_hashes:n` `__hook_double_hashes:w` `__hook_double_hashes_output:N` `__hook_double_hashes_stop:w` `__hook_double_hashes_group:n` `__hook_double_hashes_space:w` `__hook_double_hashes:n` loops through the token list #1 and duplicates any catcode 6 token, and expands tokens `\ifx`-equal to `\c__hook_hash_tl`, and leaves all other tokens `\notexpanded` with `\exp_not:N`. Unfortunately pairs of explicit catcode 1 and catcode 2 character tokens are normalised to `{1}` and `}1` because it's not feasible to expandably detect the character code (*maybe* it could be done using something along the lines of <https://tex.stackexchange.com/a/527538>, but it's far too much work for close to zero benefit).

`__hook_double_hashes:w` is the tail-recursive loop macro, that tests which of the three types of item is in the head of the token list.

```

246 \cs_new:Npn \__hook_double_hashes:n #1
247   { \__hook_double_hashes:w #1 \q__hook_recursion_tail \q__hook_recursion_stop }

```

```

248 \cs_new:Npn \__hook_double_hashes:w #1 \q__hook_recursion_stop
249   {
250     \tl_if_head_is:N_type:nTF {#1}
251       { \__hook_double_hashes_output:N }
252       {
253         \tl_if_head_is_group:nTF {#1}
254           { \__hook_double_hashes_group:n }
255           { \__hook_double_hashes_space:w }
256       }
257     #1 \q__hook_recursion_stop
258   }

```

`__hook_double_hashes_output:N` checks for the end of the token list, then checks if the token is `\c__hook_hash_tl`, and if so just leaves it.

```

259 \cs_new:Npn \__hook_double_hashes_output:N #1
260   {
261     \if_meaning:w \q__hook_recursion_tail #1
262       \__hook_double_hashes_stop:w
263     \fi:
264     \if_meaning:w \c__hook_hash_tl #1

```

(this `\use_i:nnnn` uses `\fi:` and consumes `\use:n`, the whole `\if_catcode:w` block, and the `\exp_not:N`, leaving just `#1` which is `\c__hook_hash_tl`.)

```

265   \use_i:nnnn
266 \fi:
267 \use:n
268 {

```

If `#1` is not `\c__hook_hash_tl`, then check if its catcode is 6, and if so, leave it doubled in `\exp_not:n` and consume the following `\exp_not:N #1`.

```

269   \if_catcode:w ## \exp_not:N #1
270     \exp_after:wN \use_i:nnnn
271   \fi:
272   \use_none:n
273     { \exp_not:n { #1 #1 } }
274 }

```

If both previous tests returned `false`, then leave the token unexpanded and resume the loop.

```

275   \exp_not:N #1
276   \__hook_double_hashes:w
277 }
278 \cs_new:Npn \__hook_double_hashes_stop:w #1 \q__hook_recursion_stop { \fi: }

```

Dealing with spaces and grouped tokens is trivial:

```

279 \cs_new:Npn \__hook_double_hashes_group:n #1
280   { { \__hook_double_hashes:n {#1} } \__hook_double_hashes:w }
281 \exp_last_unbraced:NNo
282 \cs_new:Npn \__hook_double_hashes_space:w \c_space_tl
283   { ~ \__hook_double_hashes:w }

```

(End definition for `__hook_double_hashes:n` and others.)

4.5.2 Patching by retokenization

At this point we've drained the possibilities of patching a command by expansion-and-redefinition, so we have to resort to patching by retokenizing the command. Patching by retokenization is done by getting the `\meaning` of the command, doing the necessary manipulations on the generated string, and the retokenizing that again by using `\scantokens`.

Patching by retokenization is definitely a riskier business, because it relies that the tokens printed by `\meaning` produce the exact same tokens as the ones in the original definition. That is, the catcode régime must be exactly(ish) the same, and there is no way of telling except by trial and error.

`__hook_retokenize_patch:Nnn` This is the macro that will control the whole process. First we'll try out one final, rather trivial case, of a command with no arguments; that is, a token list. This case can be patched with the expand-and-redefine routine but it has to be the very last case tested for, because most (all?) robust commands start with a top-level macro with no arguments, so testing this first would short-circuit `\robust@command@act` and the top-level macros would be incorrectly patched. In that case, we just check if the `\cs_argument_spec:N` is empty, and call `__hook_patch_expand_redefine>NNnn`.

```
284 \cs_new_protected:Npn \__hook_retokenize_patch:Nnn #1 #2 #3
285   {
286     \__hook_patch_debug:x { ..~command~can~only~be~patched~by~rescanning }
287     \str_if_eq:eeTF { \cs_argument_spec:N #1 } { }
288       { \__hook_patch_expand_redefine:NNnn \c_false_bool #1 {#2} {#3} }
289   }
```

Otherwise, we start the actual patching by retokenization job. The code calls `__hook_try_patch_with_catcodes:Nnnnw` with a different catcode setting:

- The current catcode setting;
- Switching the catcode of `\@`;
- Switching the `\Expl3` syntax on or off;
- Both of the above.

If patching succeeds, `__hook_try_patch_with_catcodes:Nnnnw` has the side-effect of patching the macro `#1` (which may be an internal from the command whose name is `#2`).

```
290   \tl_set:Nx \l__hook_tmpa_tl
291   {
292     \int_compare:nNnTF { \char_value_catcode:n {'\@} } = { 12 }
293       { \exp_not:N \makeatletter } { \exp_not:N \makeatother }
294   }
295   \tl_set:Nx \l__hook_tmpb_tl
296   {
297     \bool_if:NTF \l__kernel_expl_bool
298       { \ExplSyntaxOff } { \ExplSyntaxOn }
299   }
300   \use:x
301   {
302     \exp_not:N \__hook_try_patch_with_catcodes:Nnnnw
303       \exp_not:n { #1 {#2} {#3} }
304       { \prg_do_nothing: }
```

```

305     { \exp_not:V \l__hook_tmpa_tl } % @
306     { \exp_not:V \l__hook_tmpb_tl } % _:
307     {
308         \exp_not:V \l__hook_tmpa_tl    % @
309         \exp_not:V \l__hook_tmpb_tl    % _:
310     }
311 }
312 \q_recursion_tail \q_recursion_stop

```

If no catcode setting succeeds, give up and raise an error. The command isn't changed in any way in that case.

```

313 {
314     \msg_error:n{nnxx}{hooks}{cant-patch}
315     { \c_backslash_str #2 } { retok }
316 }
317 }
318 }

```

(End definition for `_hook_retokenize_patch:Nnn`.)

This function is a simple wrapper around `_hook_cmd_if_scannable:NnTF` and `_hook_patch_retokenize:Nnn` if the former returns `<true>`, plus some debug messages.

```

319 \cs_new_protected:Npn \_hook_try_patch_with_catcodes:Nnnw #1 #2 #3 #4
320 {
321     \quark_if_recursion_tail_stop_do:nn {#4} { \use:n }
322     \_hook_patch_debug:x { ++trying-to-patch-by-retokenization }
323     \_hook_cmd_if_scannable:NnTF {#1} {#4}
324     {
325         \_hook_patch_debug:x { +macro-can-be-retokenized-cleanly }
326         \_hook_patch_debug:x { ==retokenizing-macro-now }
327         \_hook_patch_retokenize:Nnn #1 {#2} {#3} {#4}
328         \use_i_delimit_by_q_recursion_stop:nw \use_none:n
329     }
330     {
331         \_hook_patch_debug:x { ---macro-cannot-be-retokenized-cleanly }
332         \_hook_try_patch_with_catcodes:Nnnw #1 {#2} {#3}
333     }
334 }

```

(End definition for `_hook_try_patch_with_catcodes:Nnnw`.)

`\kerneltmpDoNotUse` This is an oddity required to be safe (as safe as reasonably possible) when patching the command. The entirety of

`<prefixes> \def <cs> <parameter text> {<replacement text>}`

will go through `\scantokens`. The `<parameter text>` and `<replacement text>` are what we are trying to retokenize, so not much worry there. The other items, however, should "just work", so some care is needed to not use too fancy catcode settings. Therefore we can't use an `\Expl3`-named macro for `<cs>`, nor the `\Expl3` versions of `\def` or the `<prefixes>`. That is why the definitions that will eventually go into `\scantokens` will use the oddly (but hopefully clearly)-named `\kerneltmpDoNotUse`:

```
335 \cs_new_eq:NN \kerneltmpDoNotUse !
```

PhO: Maybe this can be avoided by running the <parameter text> and the <replacement text> separately through \scantokens and then putting everything together at the end.

(End definition for \kerneltmpDoNotUse.)

__hook_patch_required_catcodes: Here are the catcode settings that are *mandatory* when retokenizing commands. These are the minimum necessary settings to perform the definitions: they identify control sequences, which must be escaped with \backslash_0 , delimit the definition with $\{_1$ and $\}_2$, and mark parameters with $\#_6$. Everything else may be changed, but not these.

```
336 \cs_new_protected:Npn \_\_hook_patch_required_catcodes:
337 {
338     \char_set_catcode_escape:N \\ 
339     \char_set_catcode_group_begin:N \{
340     \char_set_catcode_group_end:N \}
341     \char_set_catcode_parameter:N \#
342     % \int_set:Nn \tex_endlinechar:D { -1 }
343     % \int_set:Nn \tex_newlinechar:D { -1 }
344 }
```

PhO: etoolbox sets the `\endlinechar` and `\newlinechar` when patching, but as far as I tested these didn't make much of a difference, so I left them out for now. Maybe `\newlinechar=-1` avoids a space token being added after the definition.

PhO: If the patching is split by `\UseHook{cmd/foobefore}`, all the tokens there need to have the right catcodes, so this list now includes all lowercase letters, U and H, the slash, and whatever characters in the command name... sigh...

(End definition for __hook_patch_required_catcodes:.)

__hook_cmd_if_scorable:Nn~~TF~~: Here we'll do a quick test if the command being patched can in fact be retokenized with the specific catcode setting without changing in meaning. The test is straightforward:

1. apply `\meaning` to the command;
2. split the `\def\kerneltmpDoNotUse{parameter text}{replacement text}` and arrange them as
`\def\kerneltmpDoNotUse{parameter text}{replacement text}`
3. rscan that with the given catcode settings, and do the definition; then finally
4. compare `\kerneltmpDoNotUse` with the original command.

If both are `\ifx`-equal, the command can be safely patched.

```
345 \prg_new_protected_conditional:Npnn \_\_hook_cmd_if_scorable:Nn #1 #2 { TF }
346 {
347     \cs_set_eq:NN \kerneltmpDoNotUse \scan_stop:
348     \cs_set_eq:NN \_\_hook_tmp:w \scan_stop:
349     \use:x
350     {
351         \cs_set:Npn \_\_hook_tmp:w
352             #####1 \tl_to_str:n { macro: } #####2 -> #####3 \s__hook_mark
353             { #####1 \def \kerneltmpDoNotUse #####2 {#####3} }
354         \tl_set:Nx \exp_not:N \l__hook_tmpa_tl
355             { \exp_not:N \_\_hook_tmp:w \token_to_meaning:N #1 \s__hook_mark }
356     }
357     \tl_rescan:nV { #2 \_\_hook_patch_required_catcodes: } \l__hook_tmpa_tl
358     \token_if_eq_meaning:NNTF #1 \kerneltmpDoNotUse
```

```

359      { \prg_return_true: }
360      { \prg_return_false: }
361  }

```

(End definition for `_hook_cmd_if_scansable:NnTF`.)

`_hook_patch_retokenize:Nnnn` Then, if `_hook_cmd_if_scansable:NnTF` returned true, we can go on and patch the command.

```

362 \cs_new_protected:Npn \_hook_patch_retokenize:Nnnn #1 #2 #3 #4
363 {

```

Start off by making some things `\relax` to avoid lots of `\noexpand` below.

```

364 \cs_set_eq:NN \kerneltmpDoNotUse \scan_stop:
365 \cs_set_eq:NN \_hook_tmp:w \scan_stop:
366 \use:x
367 {

```

Now we'll define `_hook_tmp:w` such that it splits the `\meaning` of the macro (#1) into its three parts:

```

#####1. <prefixes>
#####2. <parameter text>
#####3. <replacement text>

```

and arrange that a complete definition, then place the `before` or `after` hooks around the `<replacement text>`: accordingly.

```

368 \cs_set:Npn \_hook_tmp:w
369     #####1 \tl_to_str:n { macro: } #####2 -> #####3 \s__hook_mark
370 {
371     #####1 \def \kerneltmpDoNotUse #####
372         {
373             \str_if_eq:nnT {#3} { before }
374                 { \token_to_str:N \UseHook { cmd / #2 / #3 } }
375             #####
376             \str_if_eq:nnT {#3} { after }
377                 { \token_to_str:N \UseHook { cmd / #2 / #3 } }
378         }
379     }

```

Now we just have to get the `\meaning` of the command being patched and pass it through the meat grinder above.

```

380 \tl_set:Nx \exp_not:N \l__hook_tmpa_tl
381     { \exp_not:N \_hook_tmp:w \token_to_meaning:N #1 \s__hook_mark }
382 }

```

Now rescan with the given catcode settings (overridden by the `_hook_patch_required_catcodes:`), and implicitly (by using the rescanned token list) carry out the definition from above.

```
383 \tl_rescan:Nv { #4 \_hook_patch_required_catcodes: } \l__hook_tmpa_tl
```

And to close, copy the newly-defined command into the old name and the patching is finally completed:

```

384 \cs_gset_eq:NN #1 \kerneltmpDoNotUse
385 }

```

(End definition for `_hook_patch_retokenize:Nnnn`.)

4.6 Messages

```

386 <latexrelease>\IncludeInRelease{2021/11/15}{wrong-cmd-hook}%
387 <latexrelease>                                {Standardise-generic-hook-names}
388 <latexrelease>\EndIncludeInRelease
389 <latexrelease>\IncludeInRelease{2021/11/15}{wrong-cmd-hook}%
390 <latexrelease>                                {Standardise-generic-hook-names}
391 <latexrelease>\msg_new:nnn { hooks } { wrong-cmd-hook }
392 <latexrelease>  {
393 <latexrelease>    Generic~hook~`cmd/#1/#2'~is~invalid.
394 <latexrelease>%     The~hook~should~be~`cmd/#1/before'~or~`cmd/#1/after'.
395 <latexrelease>  }
396 <latexrelease>  {
397 <latexrelease>    You~tried~to~add~a~generic~hook~to~command~\iow_char:N \\#1,~but~`#2'~
398 <latexrelease>    is~an~invalid~component.~Only~`before'~or~`after'~are~allowed.
399 <latexrelease>  }
400 <latexrelease>\EndIncludeInRelease
401 \msg_new:nnn { hooks } { cant-patch }
402   {
403     Generic~hooks~cannot~be~added~to~`#1'.
404   }
405   {
406     You~tried~to~add~a~hook~to~`#1',~but~LaTeX~was~unable~to~
407     patch~the~command~because~it~\_\_hook\_unpatchable\_cases:n {#2}.
408   }
409 \cs_new:Npn \_\_hook_unpatchable_cases:n #1
410   {
411     \str_case:nn {#1}
412       {
413         { undef } { doesn't-exist }
414         { macro } { is-not-a-macro }
415         { expl3 } { is-a-private-expl3-macro }
416         { retok } { can't-be-retokenized-cleanly }
417       }
418   }
419 <latexrelease>\IncludeInRelease{0000/00/00}{ltcmandhooks}%
420 <latexrelease>                                {The~hook~management~system~for~commands}
421 <latexrelease>

```

The command `__hook_cmd_begindocument_code:` is used in an internal hook, so we need to make sure it has a harmless definition after rollback as that will not remove it from the kernel hook.

```

422 <latexrelease>\cs_set_eq:NN \_\_hook_cmd_begindocument_code: \prg_do_nothing:
423 <latexrelease>
424 <latexrelease>\EndModuleRelease
425 \ExplSyntaxOff
426 </2ekernel | latexrelease>
427 <@@=〉

```

File j

ltalloc.dtx

1 Counters

This section deals with counter and other variable allocation.

1 `(*2ekernel)`

The following are from plain TEX:

\z@ A zero dimen or number. It's more efficient to write `\parindent\z@` than `\parindent 0pt`.

\cne The number 1.

\m@cne The number -1.

\tw@ The number 2.

\sixt@@n The number 16.

\c@m The number 1000.

\c@MM The number 20000.

\c@xxxii The constant 32.

2 `\chardef\c@xxxii=32`

(End definition for \c@xxxii.)

\c@Mi Constants 10001–10004.

\c@mii 3 `\mathchardef\c@Mi=10001`

\c@miii 4 `\mathchardef\c@Mii=10002`

\c@miv 5 `\mathchardef\c@Miii=10003`

6 `\mathchardef\c@Miv=10004`

(End definition for \c@Mi and others.)

\c@tempcnta Scratch count registers used by LATEX kernel commands.

\c@tempcntb 7 `\newcount\c@tempcnta`

8 `\newcount\c@tempcntb`

(End definition for \c@tempcnta and \c@tempcntb.)

\c@if@tempswa General boolean switch used by LATEX kernel commands.

9 `\newif\c@if@tempswa`

(End definition for \c@if@tempswa.)

\c@tempdima Scratch dimen registers used by LATEX kernel commands.

\c@tempdimb 10 `\newdimen\c@tempdima`

\c@tempdimc 11 `\newdimen\c@tempdimb`

12 `\newdimen\c@tempdimc`

(End definition for \c@tempdima, \c@tempdimb, and \c@tempdimc.)

\@tempboxa Scratch box register used by L^AT_EX kernel commands.
 ¹³ \newbox\@tempboxa
 (End definition for \@tempboxa.)

\@tempskipa Scratch skip registers used by L^AT_EX kernel commands.
 \@tempskipb
 ¹⁴ \newskip\@tempskipa
 ¹⁵ \newskip\@tempskipb
 (End definition for \@tempskipa and \@tempskipb.)

\@temptokena Scratch token register used by L^AT_EX kernel commands.
 ¹⁶ \newtoks\@temptokena
 (End definition for \@temptokena.)

\@flushglue Glue used for \right- & \leftskip = 0pt plus 1fil
 ¹⁷ \newskip\@flushglue \@flushglue = 0pt plus 1fil
 (End definition for \@flushglue.)
 ¹⁸ ⟨/2ekernel⟩

File k

ltcntrl.dtx

1 Program control structure

This section defines a number of control structure macros, such as while-loops and for-loops.

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
1  {*2ekernel}
2  \message{control,}

\@whilenum TEST \do {BODY}
\@whiledim TEST \do {BODY} : These implement the loop
    while TEST do BODY od
    where TEST is a TeX \ifnum or \ifdim test, respectively.
    They are optimized for the normal case of TEST initially false.

\@whilesw SWITCH \fi {BODY} : Implements the loop
    while SWITCH do BODY od
    Optimized for normal case of SWITCH initially false.

\@for NAME := LIST \do {BODY} : Assumes that LIST expands to A1,A2,
... ,An .
    Executes BODY n times, with NAME = Ai on the i-th iteration.
    Optimized for the normal case of n = 1. Works for n=0.

\@tfour NAME := LIST \do {BODY}
    if, before expansion, LIST = T1 ... Tn where each Ti is a
    token or {...}, then executes BODY n times, with NAME = Ti
    on the i-th iteration. Works for n=0.
```

NOTES: 1. These macros use no `\@temp` sequences.
2. These macros do not work if the body contains anything that looks syntactically to TeX like an improperly balanced `\if` `\else` `\fi`.

```
\@whilenum TEST \do {BODY} ==
BEGIN
    if TEST
        then BODY
            \@iwhilenum{TEST \relax BODY}
END

\@iwhilenum {TEST BODY} ==
BEGIN
    if TEST
        then BODY
```

```

        \cnextwhile = def(\@iwhilenum)
else  \cnextwhile = def(\@whilenoop)
fi
\cnextwhile {TEST BODY}
END

\@whilesw SWITCH \fi {BODY} ==
BEGIN
if SWITCH
then BODY
\@iwhilesw {SWITCH BODY}\fi
fi
END

\@iwhilesw {SWITCH BODY} \fi ==
BEGIN
if SWITCH
then BODY
\cnextwhile = def(\@iwhilesw)
else \cnextwhile = def(\@whileswnoop)
fi
\cnextwhile {SWITCH BODY} \fi
END

```

End of historical L^AT_EX 2.09 comments.

```

\@whilenoop
\@whilenum
\@iwhilenum
3 \long\def\@whilenum#1\do #2{\ifnum #1\relax #2\relax\@iwhilenum{#1\relax
4      #2\relax}\fi}
5 \long\def\@iwhilenum#1{\ifnum #1\expandafter\@iwhilenum
6      \else\expandafter\@gobble\fi{#1}}

```

(End definition for \@whilenoop, \@whilenum, and \@iwhilenum.)

```

\@whiledim
\@iwhiledim
7 \long\def\@whiledim#1\do #2{\ifdim #1\relax#2\@iwhiledim{#1\relax#2}\fi}
8 \long\def\@iwhiledim#1{\ifdim #1\expandafter\@iwhiledim
9      \else\expandafter\@gobble\fi{#1}}

```

(End definition for \@whiledim and \@iwhiledim.)

```

\@whileswnoop
\@whilesw
\@iwhilesw
10 \long\def\@whilesw#1\fi{#1#2\@iwhilesw{#1#2}\fi\fi}
11 \long\def\@iwhilesw#1\fi{#1\expandafter\@iwhilesw
12      \else\@gobbletwo\fi{#1}\fi}

```

(End definition for \@whilesnoop, \@whilesw, and \@iwhilesw.)

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
\@for NAME := LIST \do {BODY} ==
  BEGIN \@forloop expand(LIST),\@nil,\@nil \@@ NAME {BODY} END

\@forloop CAR, CARCDR, CDRCDR \@@ NAME {BODY} ==
  BEGIN
    NAME = CAR
    if def(NAME) = def(\@nnil)
      else BODY;
      NAME = CARCDR
      if def(NAME) = def(\@nnil)
        else BODY
          \@iforloop CDRCDR \@@ NAME \do {BODY}
        fi
      fi
  END

\@iforloop CAR, CDR \@@ NAME {BODY} =
  NAME = CAR
  if def(NAME) = def(\@nnil)
    then \@nextwhile = def(\@fornoop)
    else BODY ;
      \@nextwhile = def(\@iforloop)
  fi
  \@nextwhile name cdr {body}

\@tfor NAME := LIST \do {BODY}
  = \@tforloop LIST \@nil \@@ NAME {BODY}

\@tforloop car cdr \@@ name {body} =
  name = car
  if def(name) = def(\@nnil)
    then \@nextwhile == \@fornoop
    else body ;
      \@nextwhile == \@forloop
  fi
  \@nextwhile name cdr {body}
```

End of historical L^AT_EX 2.09 comments.

\@nnil

¹³ \def\@nnil{\@nil}

(End definition for \@nnil.)

\@empty

¹⁴ \def\@empty{}

(End definition for \@empty.)

```

\@fornoop
15 \long\def\@fornoop#1\@@#2#3{%
(End definition for \@fornoop.)}

\@for
16 \long\def\@for#1:=#2\do#3{%
17   \expandafter\def\expandafter\@fortmp\expandafter{#2}%
18   \ifx\@fortmp\empty \else
19     \expandafter\@forloop#2,\@nil,\@nil\@@#1{#3}\fi}
(End definition for \@for.)

\@forloop
20 \long\def\@forloop#1,#2,#3\@@#4#5{\def#4{#1}\ifx #4\@nnil \else
21   #5\def#4{#2}\ifx #4\@nnil \else#5\@iforloop #3\@@#4{#5}\fi\fi}
(End definition for \@forloop.)

\@iforloop
22 \long\def\@iforloop#1,#2\@@#3#4{\def#3{#1}\ifx #3\@nnil
23   \expandafter\@fornoop \else
24     #4\relax\expandafter\@iforloop\fi#2\@@#3{#4}}
(End definition for \@iforloop.)

\@tfor
25 \def\@tfor#1:={\@tfctr#1 }
26 \long\def\@tfctr#1#2\do#3{\def\@fortmp{#2}\ifx\@fortmp\space\else
27   \@tforloop#2\@nil\@nil\@@#1{#3}\fi}
28 \long\def\@tforloop#1#2\@@#3#4{\def#3{#1}\ifx #3\@nnil
29   \expandafter\@fornoop \else
30     #4\relax\expandafter\@tforloop\fi#2\@@#3{#4}}
(End definition for \@tfor.)

\@break@tfor Break out of a \@tfor loop. This should be called inside the scope of an \if. See
\@iffilenameonpath for an example.
31 \long\def\@break@tfor#1\@@#2#3{\fi\fi}
(End definition for \@break@tfor.)

\@removeelement Removes an element from a comma-separated list and puts it into a control se-
quence, called as \@removeelement{\langle element\rangle}{\langle list\rangle}{\langle cs\rangle}. Due to the implemen-
tation method the \langle element\rangle is not allowed to contain braces.
32 \def\@removeelement#1#2#3{%
33   \def\reserved@a##1,#1,##2\reserved@a{##1,##2\reserved@b}%
34   \def\reserved@b##1,\reserved@b##2\reserved@b{%
35     \ifx,##1\empty\else##1\fi}%
36   \edef#3{%
37     \expandafter\reserved@b\reserved@a,#2,\reserved@b,#1,\reserved@a}}
(End definition for \@removeelement.)
38 ⟨/2ekernel⟩

```

File 1

lterror.dtx

1 Error handling and tracing

This section defines L^AT_EX's error commands.

```
1 (*2ekernel)
```

The ‘2ekernel’ code ensures that a \usepackage{autoerr} is essentially ignored if a ‘full’ format is being used that has the error messages already in the format.

These days we don't support autoloading approach any longer, but this part bit is kept in case it is used in old documents.

```
2 \expandafter\let\csname ver@autoerr.sty\endcsname\fmtversion
```

1.1 General commands

\MessageBreak This command prints a new-line inside a message, followed by a continuation line begun with \cmsg@continuation. Normally it is defined to be \relax, but inside messages, it is let to \message@break.

```
3 \let\MessageBreak\relax
```

(End definition for \MessageBreak.)

\GenericInfo This takes two arguments: a continuation and a message, and sends the result to the log file.

```
4 \DeclareRobustCommand{\GenericInfo}[2]{%
5   \begingroup
6     \def\MessageBreak{^^J#1}%
7     \set@display@protect
8     \immediate\write\m@ne{#2\on@line.}%
9   \endgroup
10 }
```

(End definition for \GenericInfo.)

\GenericWarning This takes two arguments: a continuation and a message, and sends the result to the screen.

```
11 \DeclareRobustCommand{\GenericWarning}[2]{%
12   \begingroup
13     \def\MessageBreak{^^J#1}%
14     \set@display@protect
15     \immediate\write\@unused{^^J#2\on@line.^^J}%
16   \endgroup
17 }
```

(End definition for \GenericWarning.)

- \GenericError This macro takes four arguments: a continuation, an error message, where to go for further information, and the help information. It displays the error message, and sets the error help (the result of typing `h` to the prompt), and does a horrible hack to turn the last context line (which by default is the only context line) into just three dots. This could be made more efficient.

```

18 \bgroup
19 \lccode`\@`\
20 \lccode`\-=`\
21 \lccode`\)=`\
22 \lccode`\{`\
23 \lccode`\T`=\T%
24 \lccode`\H`=\H%
25 \catcode`\ =11\relax%
26 \lowercase{%
27 \egroup%

```

Unfortunately TeX versions older than 3.141 have a bug which means that `^^J` does not force a linebreak in `\message` and `\errmessage` commands. So for these old TeX's we use `\typeout` to produce the message, and then have an empty `\errmessage` command. This causes an extra line of the form

To appear on the terminal, but if you do not like it, you can always upgrade your TeX! In order for your format to use this version, you must define the macro `\@TeXversion` to be the version number, e.g., 3.14 of the underlying TeX. See the comments in `ltdircheck.dtx`.

```

28 \dimen@\ifx\@TeXversion\undefined\else\@TeXversion\fi\p@%
29 \ifdim\dimen@>3.14\p@%

```

First the ‘standard case’.

```

30 \DeclareRobustCommand{\GenericError}[4]{%
31 \begingroup%
32 \immediate\write\@unused{()}%
33 \def\MessageBreak{^^J}%
34 \set@display@protect%
35 \edef%
36 %   %<-----do not delete this space!----->%
37 \err@ %
38 {{#4}}%
39 \errhelp%
40 %   %<-----do not delete this space!----->%
41 \err@ %
42 \let%
43 %   %<-----do not delete this space!----->%
44 \err@ %
45 \empty%
46 \def\MessageBreak{^^J#1}%
47 \def~{\errmessage{%
48 #2.^^J^^J%
49 #3^^J%
50 Type H <return> for immediate help%
51 %   %<-----do not delete this space!----->%
52 \err@ %

```

```

53  } } %
54  ~%
55  \endgroup}%
56  \else%
      Secondly the version for old TEX's.
57  \DeclareRobustCommand{\GenericError}[4]{%
58  \begingroup%
59  \immediate\write\@unused{ }%
60  \def\MessageBreak{^ }%
61  \set@display@protect%
62  \edef%
63  %   %<-----do not delete this space!----->%
64  \err@ %
65  {{#4}}%
66  \errhelp%
67  %   %<-----do not delete this space!----->%
68  \err@ %
69  \let%
70  %   %<-----do not delete this space!----->%
71  \err@ %
72  \errmessage%
73  \def\MessageBreak{^ J#1}%
74  \def~{\typeout{! } %
75  #2.^ J^ J%
76  #3^ J%
77  Type H <return> for immediate help.)%
78  %   %<-----do not delete this space!----->%
79  \err@ %
80  {}}%
81  ~%
82  \endgroup}%
83  \fi}%

```

(End definition for `\GenericError`.)

<code>\PackageError</code> <code>\PackageWarning</code> <code>\PackageWarningNoLine</code> <code>\PackageInfo</code> <code>\ClassError</code> <code>\ClassWarning</code> <code>\ClassWarningNoLine</code> <code>\ClassInfo</code>	These commands are intended for use by package and class writers, to give information to authors. The syntax is: <pre> \PackageError{<package>}{<error>}{<help>} \PackageWarning{<package>}{<warning>} \PackageWarningNoLine{<package>}{<warning>} \PackageInfo{<package>}{<info>} </pre>
--	---

and similarly for classes. The `Error` commands print the `<error>` message, and present the interactive prompt; if the author types `h`, then the `<help>` information is displayed. The `Warning` commands produce a warning but do not present the interactive prompt. The `WarningNoLine` commands do the same, but don't print the input line number. The `Info` commands write the message to the `log` file. Within the messages, the command `\MessageBreak` can be used to break a line, `\protect` can be used to protect command names, and `\space` is a space, for example:

```
\newcommand{\foo}{FOO}
\PackageWarning{ethel}{%
    Your hovercraft is full of eels,\MessageBreak
    and \protect\foo\space is \foo}
```

produces:

```
Package ethel warning: Your hovercraft is full of eels,
(ethel)                                and \foo is FOO on input line 54.

84 \gdef\PackageError#1#2#3{%
85   \GenericError{%
86     (#1)\@spaces\@spaces\@spaces\@spaces
87   }{%
88     Package #1 Error: #2%
89   }{%
90     See the #1 package documentation for explanation.%}
91   }{#3}%
92 }

93 \def\PackageWarning#1#2{%
94   \GenericWarning{%
95     (#1)\@spaces\@spaces\@spaces\@spaces
96   }{%
97     Package #1 Warning: #2%
98   }{%
99 }
100 \def\PackageWarningNoLine#1#2{%
101   \PackageWarning{#1}{#2\@gobble}%
102 }
103 \def\PackageInfo#1#2{%
104   \GenericInfo{%
105     (#1) \@spaces\@spaces\@spaces
106   }{%
107     Package #1 Info: #2%
108   }{%
109 }
110 \gdef\ClassError#1#2#3{%
111   \GenericError{%
112     (#1) \space\@spaces\@spaces\@spaces
113   }{%
114     Class #1 Error: #2%
115   }{%
116     See the #1 class documentation for explanation.%}
117   }{#3}%
118 }

119 \def\ClassWarning#1#2{%
120   \GenericWarning{%
121     (#1) \space\@spaces\@spaces\@spaces
122   }{%
123     Class #1 Warning: #2%
124   }{%
125 }
126 \def\ClassWarningNoLine#1#2{%
```

```

127      \ClassWarning{#1}{#2\@gobble}%
128  }
129  \def\ClassInfo#1#2{%
130      \GenericInfo{%
131          (#1) \space\space\@spaces\@spaces
132      }{%
133          Class #1 Info: #2%
134      }%
135  }

```

(End definition for `\PackageError` and others.)

```

\ClassNote
\ClassNoteNoLine 136 </2ekernel>
\PackageNote 137 <*2ekernel | latexrelease>
\PackageNoteNoLine 138 <latexrelease>\IncludeInRelease{2021/11/15}%
139 <latexrelease>          {\ClassNote}{Notes for classes/packages}%
\def\ClassNote#1#2{%
141      \GenericWarning{%
142          (#1) \space\space\@spaces\@spaces
143      }{%
144          Class #1 Info: #2%
145      }%
146  }
147  \def\ClassNoteNoLine#1#2{\ClassNote{#1}{#2\@gobble}}
\def\PackageNote#1#2{%
148      \GenericWarning{%
149          (#1) \@spaces\@spaces\@spaces
150      }{%
151          Package #1 Info: #2%
152      }%
153  }
154  \def\PackageNoteNoLine#1#2{\PackageNote{#1}{#2\@gobble}}
155 </2ekernel | latexrelease>
156 <latexrelease>\EndIncludeInRelease
157

```

We don't roll back, because if this code is used by packages then most often they will not have rollback code implemented, so they would immediately break even if they otherwise would work fine.

```

158 <latexrelease>\IncludeInRelease{0000/00/00}%
159 <latexrelease>          {\ClassNote}{Notes for classes/packages}%
160 <latexrelease>
161 <latexrelease>\EndIncludeInRelease
162 <*2ekernel>

```

(End definition for `\ClassNote` and others.)

`\@latex@error` Errors and other info, for use in the L^AT_EX core.

```

\@latex@warning 163 \gdef\@latex@error#1#2{%
164     \GenericError{%
165         \space\space\space\@spaces\@spaces\@spaces
166     }{%
167         LaTeX Error: #1%
168     }{%

```

```

169      See the LaTeX manual or LaTeX Companion for explanation.%  

170  }{#2}%
171 }
172 \def\@latex@warning#1{%
173   \GenericWarning{%
174     \space\space\space\@spaces\@spaces\@spaces
175   }{%
176     LaTeX Warning: #1%
177   }%
178 }
179 \def\@latex@warning@no@line#1{%
180   \@latex@warning{#1\@gobble}}
181 \def\@latex@info#1{%
182   \GenericInfo{%
183     \@spaces\@spaces\@spaces
184   }{%
185     LaTeX Info: #1%
186   }%
187 }
188 \def\@latex@info@no@line#1{%
189   \@latex@info{#1\@gobble}}

```

\@font@warning and \@font@info are defined later since they have to be redefined by the `traceint` package.

```

def\@font@warning#1{%
  \GenericWarning{%
    {(font)}\@spaces\@spaces}%
    {Font Warning: #1}%
}
def\@font@info#1{%
  \GenericInfo{%
    (font)\space\@spaces
  }{%
    Font Info: #1%
  }%
}

```

(End definition for \@latex@error and others.)

\@latex@note These are “info” messages that display on the terminal not just in the transcript.

```

190  (/2ekernel)
191  {*2ekernel | latexrelease}
192  <latexrelease>\IncludeInRelease{2021/11/15}%
193  <latexrelease>          {\@latex@note}{Display notes}%
194 \def\@latex@note#1{%
195   \GenericWarning{%
196     \@spaces\@spaces\@spaces
197   }{%
198     LaTeX Info: #1%
199   }%
200 }

```

```

201 \def\@latex@note@no@line#1{%
202   \@latex@note{#1\@gobble}}

```

We don't make them undefined but rather point to `\@latex@info` because that's what they replace. This way we can change `\@latex@info` elsewhere without the need to further rollback sections.

```

203 </2ekernel | latexrelease>
204 <latexrelease>\EndIncludeInRelease
205 <latexrelease>\IncludeInRelease{0000/00/00}%
206 <latexrelease>           {\@latex@note}{Display notes}%
207 <latexrelease>
208 <latexrelease>\let\@latex@note\@latex@info
209 <latexrelease>\let\@latex@note@no@line\@latex@info@no@line
210 <latexrelease>\EndIncludeInRelease
211 <*2ekernel>

```

(*End definition for `\@latex@note` and `\@latex@note@no@line`.*)

`\c@errorcontextlines` `\errorcontextlines` as a L^AT_EX counter, so that it may be manipulated with `\setcounter` (once it is defined :-)

```

212 \let\c@errorcontextlines\errorcontextlines
213 \c@errorcontextlines=-1

```

(*End definition for `\c@errorcontextlines`.*)

`\on@line` The message ‘on input line *n*’.

```

214 \def\on@line{ on input line \the\inputlineno}

```

(*End definition for `\on@line`.*)

`\@warning` `\@@warning` Older L^AT_EX messages. For the moment, these `\let` to the new message commands. They may be changed later, once only obsolete packages and classes contain them.

```

215 \let\@warning\@latex@warning
216 \let\@@warning\@latex@warning@no@line
217 \global\let\@latexerr\@latex@error

```

(*End definition for `\@warning`, `\@@warning`, and `\@latexerr`.*)

`\@spaces` Four spaces.

```

218 \def\@spaces{\space\space\space\space}

```

(*End definition for `\@spaces`.*)

1.2 Specific errors

`\@eha` The more common error help messages.

```

219 \gdef\@ehaf{%
220   Your command was ignored.\MessageBreak
221   Type \space I <command> <return> \space to replace it %
222   with another command,\MessageBreak
223   or \space <return> \space to continue without it.}
224 \gdef\@ehbf{%
225   You've lost some text. \space \@ehc}
226 \gdef\@ehcf{%
227   Try typing \space <return> %

```

```

228  \space to proceed.\MessageBreak
229  If that doesn't work, type \space X <return> \space to quit.}
230  \gdef\@ehd{%
231    You're in trouble here. \space\@ehc}

```

(End definition for \@eha and others.)

- \@notdefinable Error message generated in \@ifdefinable from calls to one of the commands \newcommand, \newlength or \newtheorem specifying an already-defined command name or one that begins \end....
- ```

232 \gdef\@notdefinable{%
233 \@latex@error{%
234 Command \backslashreserved@a\space
235 already defined.\MessageBreak
236 Or name \backslashqend... illegal,
237 see p.192 of the manual}\@eha}

```

(End definition for \@notdefinable.)

- \@nolnerr Generated by \newline and \\ when called in vertical mode.

```

238 \gdef\@nolnerr{%
239 \@latex@error{There's no line here to end}\@eha}

```

(End definition for \@nolnerr.)

- \@nocounterr Generated by \setcounter, \addtocounter or \newcounter if applied to an undefined counter  $\langle cnt \rangle$ .

\@nocnterr Obsolete error message generated in L<sup>A</sup>T<sub>E</sub>X2.09 by \setcounter, \addtocounter or \newcounter for undefined counter. DO NOT use for L<sup>A</sup>T<sub>E</sub>X2 <sub>$\varepsilon$</sub>  it MIGHT vanish! Use \@nocounterr{\mathit{cnt}} instead.

```

240 \gdef\@nocnterr#1{%
241 \@latex@error{No counter '#1' defined}\@eha}
242 \gdef\@nocnterr{\@nocounterr?}

```

(End definition for \@nocounterr and \@nocnterr.)

- \@ctrerr Called when trying to print the value of a counter numbered by letters that's greater than 26.

```

243 \gdef\@ctrerr{%
244 \@latex@error{Counter too large}\@ehb}

```

(End definition for \@ctrerr.)

- \@nodocument Error produced if paragraphs are typeset in the preamble.

```

245 \gdef\@nodocument{%
246 \@latex@error{Missing \protect\begin{document}}}\@ehd}

```

(End definition for \@nodocument.)

\@badend Called by \end that doesn't match its \begin. RmS 1992/08/24: added code to \@badend to display position of non-matching \begin. FMi 1993/01/14: missing space added.

The environment name has to literally match, i.e., what is stored in \@currenvir (after one expansion) must match what is passed to \end (without expansion). If not we complain. Not the absolute best solution but at least it avoids getting \begin{foo} ended by \end{foo} which was possible in the past.

```
247 \gdef\@badend#1{%
248 \@latex@error{\protect\begin
249 {\detokenize\expandafter{\@currenvir}}\@currenvline
250 \space ended by \protect\end{\detokenize{#1}}}\@eha}
```

(End definition for \@badend.)

\@badmath Called by \[, \], \{ or \} when used in wrong mode.

```
251 \gdef\@badmath{%
252 \@latex@error{Bad math environment delimiter}\@eha}
```

(End definition for \@badmath.)

\@toodeep Called by a list environment nested more than six levels deep, or an enumerate or itemize nested more than four levels.

```
253 \gdef\@toodeep{%
254 \@latex@error{Too deeply nested}\@ehd}
```

(End definition for \@toodeep.)

\@badpoptabs Called by \endtabbing when not enough \poptabs have occurred, or by \poptabs when too many have occurred.

```
255 \gdef\@badpoptabs{%
256 \@latex@error{\protect\pushtabs\space and \protect\poptabs
257 \space don't match}\@ehd}
```

(End definition for \@badpoptabs.)

\@badtab Called by \>, \+ , \- or \< when stepping to an undefined tab.

```
258 \gdef\@badtab{%
259 \@latex@error{Undefined tab position}\@ehd}
```

(End definition for \@badtab.)

\@preamerr This error is special: it appears in places where we normally have to \protect expansions. However, to prevent a protection of the error message itself (which would result in the message getting printed not issued on the terminal) we need to locally reset \protect to \relax.

```
260 \gdef\@preamerr#1{%
261 \begingroup
262 \let\protect\relax
263 \@latex@error{\ifcase #1 Illegal character\or
264 Missing @-exp\or Missing p-arg\fi\space
265 in array arg}\@ehd
266 \endgroup}
```

(End definition for \@preamerr.)

\@badlinearg Occurs in \line and \vector command when a bad slope argument is encountered.

```
267 \gdef\@badlinearg{%
268 \@latex@error{%
269 Bad \protect\line\space or \protect\vector
270 \space argument}\@ehb}
```

(End definition for \@badlinearg.)

\@parmoderr Occurs in a float environment or a \marginpar when encountered in inner vertical mode.

```
271 \gdef\@parmoderr{%
272 \@latex@error{Not in outer par mode}\@ehb}
```

(End definition for \@parmoderr.)

\@fltovf Occurs in float environment or \marginpar when there are no more free boxes for storing floats.

```
273 \gdef\@fltovf{%
274 \@latex@error{Too many unprocessed floats}\@ehb}
```

(End definition for \@fltovf.)

\@latexbug Occurs in output routine. This is bad news.

```
275 \gdef\@latexbug{%
276 \@latex@error{This may be a LaTeX bug}{Call for help}}
```

(End definition for \@latexbug.)

\@badcrerr This error was removed and replaced by \@nolnerr.

```
277 \%def\@badcrerr {\@latex@error{Bad use of \protect\\}\@ehc}
```

(End definition for \@badcrerr.)

\@noitemerr \addvspace or \addpenalty was called when not in vmode. Probably caused by a missing \item.

```
278 \gdef\@noitemerr{%
279 \@latex@error{Something's wrong--perhaps a missing %
280 \protect\item}\@ehc}
```

(End definition for \@noitemerr.)

\@notprerr A command that can be used only in the preamble appears after the command \begin{document}.

```
281 \gdef\@notprerr{%
282 \@latex@error{Can be used only in preamble}\@eha}
```

(End definition for \@notprerr.)

\@inmatherr Issued by commands that don't work correctly within math (like \item). There is no real error recovery happening, e.g., the user might get additional errors afterwards.

```
283 \gdef\@inmatherr#1{%
284 \relax
285 \ifmmode
286 \@latex@error{Command \protect#1 invalid in math mode}\@ehc
287 \fi}
```

(End definition for \@inmatherr.)

- \@invalidchar An error for use with invalid characters. This is commented out, since we decided to use catcode 15 instead.

288 %\def\@invalidchar{\@latex@error{Invalid character in input}\@ehc}

(End definition for \@invalidchar.)

As well as the above error commands some error messages are directly coded to save space. The messages already present in L<sup>A</sup>T<sub>E</sub>X2.09 include:

Environment --- undefined

Issued by \begin for undefined environment.

Tab overflow

Occurs in \= when maximum number of tabs exceeded.

\< in mid line

Occurs in \< when it appears in middle of line.

Float(s) lost

In output routine, caused by a float environment or \marginpar occurring in inner vertical mode.

### 1.3 Tracing

The **trace** package implements the commands \traceon and \traceoff that work similar to \tracingall but skip certain code blocks that produce a lot of tracing output being of no interest during debugging (for example loading a font). Code blocks that should be hidden during tracing need to be surrounded by the macros \conditionally@traceoff and \conditionally@traceon.

For the kernel code the **trace** package then redefines a number of macros to include this tracing support.

However, in order to allow any macro package to react to \traceon we also provide dummy definitions for the two commands in the kernel so that they can be used by external packages without the need to distinguish between **trace** being loaded or not.

\conditionally@traceon These are only dummy definitions. For details see the **trace** package.

\conditionally@traceoff

289 \let\conditionally@traceon\@empty

290 \let\conditionally@traceoff\@empty

(End definition for \conditionally@traceon and \conditionally@traceoff.)

291 ⟨/2ekernel⟩

# File m

## ltpar.dtx

### 1 Paragraphs

This section of the kernel declares the commands used to set `\par` and `\everypar` whenever their function needs to be changed for a long time.

This file here describes the interfaces that have been in the kernel forever, used to implement the scenarios described below. They remain valid but are now augmented in the next file (`ltpara.dtx`) to add hooks to paragraphs. At some point we will consolidate the two files further.

There are two situations in which `\par` may be changed:

- Long-term changes, in which the new value is to remain in effect until the current environment is left. The environments that change `\par` in this way are the following:
  - All list environments (itemize, quote, etc.)
  - Environments that turn `\par` into a noop: tabbing, array and tabular.
- Temporary changes, in which `\par` is restored to its previous value the next time it is executed. The following are all such uses.
  - `\end` when preceded by `\@endparenv`, which is called by `\endtrivlist`
  - The mechanism for avoiding page breaks and getting the spacing right after section heads.

#### 1.1 Implementation

`\@setpar` To permit the proper interaction of these two situations, long-term changes are made by the `\@setpar{\VAL}` command. Its function is:

To set `\par`. It `\def`'s `\par` and `\@par` to `\VAL`.

`\@restorepar` Short-term changes are made by the usual `\def\par` commands. The original values are restored after a short-term change by the `\@restorepar` commands.

`\@@par` `\@@par` always is defined to be the original TeX `\par`.

`\everypar` `\everypar` is changed only for the short term. Whenever `\everypar` is set non-null, it should restore itself to null when executed.

The following commands change `\everypar` in this way:

- `\item`
- `\end` when preceded by `\@endparenv`, which is called by `\endtrivlist`
- `\minipage`

When dealing with `\par` and `\everypar` remember the following two warnings:

- Commands that make short-term changes to `\par` and `\everypar` must take account of the possibility that the new commands and the ones that do the restoration may be executed inside a group. In particular, `\everypar` is executed inside a group whenever a new paragraph begins with a left brace. The `\everypar` command that restores its definition should be local to the current group (in case the command is inside a minipage used inside someplace where `\everypar` has been redefined). Thus, if `\everypar` is redefined to do an `\everypar{}` it could take several executions of `\everypar` before the restoration “holds”. This usually causes no problem. However, to prevent the extra executions from doing harm, use a global switch to keep anything harmful in the new `\everypar` from being done twice.
- Commands that change `\everypar` should remember that `\everypar` might be supposed to set the following switches false:

- `@nobreak`
- `@minipage`

they should do the setting if necessary.

```

1 {*2ekernel}
2 \message{par ,}

```

`\@setpar` Initiate a long-term change to `\par`.  
`\@par`    3 `\def\@setpar#1{\def\par{\#1}\def\@par{\#1}}`

The default definition of `\@par` will ensure that if `\@restorepar` defines `\par` to execute `\@par` it will redefine itself to the primitive `\@@par` after one iteration.

```

4 \def\@par{\let\par\@@par\par}
(End definition for \@setpar and \@par.)

```

`\@restorepar` Restore from a short-term change to `\par`.  
5 `\def\@restorepar{\def\par{\@par}}`  
6 `/{/2ekernel}`

(End definition for `\@restorepar`.)

# File n

## ltpara.dtx

### Abstract

This code defines four special kernel hooks to support paragraph tagging as well as four public hooks which can be occasionally useful.

## Contents

### 1 Introduction

The building of paragraphs in the TeX engine(s) has a number of peculiarities that makes it on one hand fairly flexible but on the other hand somewhat awkward to control or reliably to extend. Thus to better understand the code below we start with a brief introduction of the mechanism; for more details refer to the TeXbook [?, chap. 14] (for the full truth you may even have to study the program code).

#### 1.1 The default processing done by the engine

TeX automatically starts building a paragraph when it is currently in vertical mode and encounters anything that can only live in horizontal mode. Most often this is a character, but there are also many commands that can be used only in horizontal mode. If any of them is encountered, TeX will immediately back up (i.e., the character or command is read later again), adds a `\parskip` glue to the current vertical list unless the list is empty, switches to horizontal mode, starts its special “start of paragraph processing” and only then rereads the character or command that caused the mode change.<sup>15</sup>

This “start of paragraph processing” first adds an empty box at the start of the horizontal list of width `\parindent` (which represents the paragraph indentation) unless the paragraph was started with `\noindent` in which case no such box is added<sup>16</sup>. It then reads and processes all tokens stored in the special engine token register `\everypar`. After that it reads and processes whatever has caused the paragraph to start.

Thus out of the box, TeX offers the possibility to put some special code into `\everypar` to gain control at (more or less) the start of the paragraph. For example, in LaTeX and a number of packages, special code like the following is sometimes used:

```
\everypar{{\setbox\z@\lastbox}\everypar{} ...}
```

This removes the paragraph indentation box again (that was already placed by TeX), then resets `\everypar` so that it doesn’t do anything on the next paragraph start and then does whatever it wants to do, e.g., in an `\item` of a list it will typeset the label in front of the paragraph text. However, there is only one such `\everypar` token register and if different packages and/or the kernel all attempt to add their own code here, coordination is very difficult if not impossible.

---

<sup>15</sup>Already not quite true: the command `\noindent` starts the paragraph but influences the special processing by suppressing the paragraph indentation box normally inserted by it.

<sup>16</sup>That’s a bit different from placing a zero-sized box!

The process when the paragraph ends has different mechanisms and interfaces. A paragraph ends when the engine primitive `\par` is called while TeX is in unrestricted horizontal mode, i.e., is building a paragraph. At other times this primitive does nothing or generates as an error depending on the mode TeX is in, e.g., the `\par` in `\hbox{a\par b}` is ignored, but `$a\par b$` would complain.

If this primitive ends the paragraph it does some special “end of horizontal list” processing, then calls TeX’s paragraph builder; this breaks the horizontal list into lines and then these lines are added as boxes to the enclosing vertical list and TeX returns to vertical mode.

This `\par` command can be given explicitly, but there are also situations in which TeX is generating it on the fly. Most often this happens when TeX encounters a blank line which is automatically changed to a `\par` command which is then executed. The other possibility is that TeX encounters a command which is incompatible with horizontal processing, e.g., `\vskip` (a request for adding vertical space). In such cases it silently backs up, and inserts a `\par` in the hope that this gets it out of horizontal mode and makes the vertical command acceptable.

The important point to note here is that TeX really inserts the command with the name `\par`, which can be redefined. Thus, it may not have its original “primitive” meaning and therefore may not end the horizontal list and call the paragraph builder. This approach offers some flexibility but also allows you to easily produce a TeX document that loops forever, for example, the simple line

```
A \let\par\relax \vskip
```

will start a horizontal list at A, redefines `\par`, then sees `\vskip` and inserts `\par` to end the paragraph. But this now only runs `\relax` so nothing changes and `\vskip` is read again, issues a `\par` which .... In short, it only takes a plain TeX document with five tokens to run forever (since no memory is consumed and therefore eventually exhausted).

There is no way other than changing `\par` to gain control at the end of a paragraph, i.e., there is no token list like `\everypar` that is inserted. Hence the only way to change the default behavior is to modify the action that `\par` executes, with similar issues as outlined before: different processes need to ensure that they do not overwrite their modifications or worse, think that the `\par` in front of them is the engine primitive while in fact it has already been changed by other code.

To make matters slightly worse there are a few places where TeX handles the situation differently (most likely for speed reasons back when computers were much slower). If TeX finds itself in unrestricted horizontal mode at the end of building a vertical box (for an `\insert`, `\vadjust` or executing the output routine code), it will finish the horizontal list not by issuing a `\par` command (which would be consistent with all other places) but by simply executing the primitive meaning of `\par`, regardless of the actual definition that `\par` has at the time.

Thus, if you have carefully crafted a redefined `\par` to execute some special actions at the end of a paragraph and you write something like

```
\vbox{Some paragraph ... text.}
```

you will find that your code does not get run for the last paragraph in that box. L<sup>A</sup>T<sub>E</sub>X avoids this problem, by making sure that its boxes (such as `\parbox` or the `minipage` environment, etc.) all internally add an explicit `\par` at the end so that such code is run and TeX finds itself in vertical mode already without the need to start up the paragraph builder internally. But, of course, this only works for boxes under direct control of the

**L**A**T**E**X** kernel; if some package uses low-level `\vboxes` without adding this precaution the **T**E**X** optimization kicks in and no special `\par` code is executed.

And there is another optimization that is painful: if a paragraph is interrupted by a mathematical display, e.g., `\[...]` in **L**A**T**E**X** or `$$...$$` in plain **T**E**X**, then **T**E**X** will resume horizontal mode afterward, i.e., it will start to build a new horizontal list without inserting an indentation box or `\everypar` at that point. However, if that list immediately ends with an explicit or implicit `\par` then **T**E**X** will simply throw away this “null” paragraph and not do its usual “end of horizontal list” processing, so this special case also needs to be accounted for when introducing any extended processing.

## 2 The new mechanism implemented for **L**A**T**E**X**

To improve the situation (and also to support automatic tagging of PDF documents) we now offer public as well as private hooks at the start and end of the paragraph processing. The public hooks can be used by packages (or by the user in the preamble or within the document) and using the hook mechanisms it is possible to reorder or arrange code from different packages in such a way that these can safely coexist.

To make that happen we have to make use of the basic functionality that is offered by **T**E**X**, e.g., we install special code inside `\everypar` to provide hooks at the beginning and we redefine `\par` to do some special processing when appropriate to install hooks at the end of the paragraph.

In order to make this work, we have to ensure that package use of `\everypar` is not overwriting our code. This is done through a trick: we basically hide the real `\everypar` from the packages and offer them a new token register (with the same name). So if they install their own code it doesn’t overwrite ours. Our code then inserts the new `\everypar` at the right place inside the process so that it looks as if it was the primitive `\everypar`.<sup>17</sup>

At the end of the paragraph it would be great if we could use a similar trick. However, due to the fact that **T**E**X** inserts the token `\par` (that doesn’t have a defined meaning) we can’t hide “the real thing<sup>TM</sup>” and offer the package an indistinguishable alternate.

Fortunately, **L**A**T**E**X** has already redefined `\par` for its own purposes. As a result there aren’t many packages that attempt to change `\par`, because without a lot of extra care that would fail miserably. But the bottom line is that, if you load a package that alters `\par` then the end of paragraph hooks are most likely not executing while that redefinition is active.<sup>18</sup>

---

<sup>17</sup>Ideally, `\everypar` wouldn’t be used at all by packages and instead they would simply write their code into the hooks now offered by the kernel. However, while this is the longterm goal and clearly an improvement (because then the packages do no longer need to worry about getting their code overwritten or needing to account for already existing code in `\everypar`), this will not happen overnight. For that reason support for this legacy method is retained.

<sup>18</sup>Similarly to the `\everypar` situation, the remedy is that such packages stop doing this and instead add their alterations into the paragraph hooks now provided.

## 2.1 The provided hooks

---

**para/before** The following four public hooks are defined and executed for each paragraph:  
**para/begin**  
**para/end**  
**para/after** **para/before** This hook is executed after the kernel hook `\@kernel@before@para@before` (discussed below) in vertical mode immediately after TEX has contributed `\parskip` to the vertical list and before the actual paragraph processing in horizontal mode starts.

This hook should either not produce any typeset material or add only vertical material. If it starts a paragraph an error is generated. The reason is that we are in the starting process of processing a paragraph and so this would lead to endless recursion.<sup>19</sup>

**para/begin** This hook is executed after the kernel hook `\@kernel@before@para@begin` (discussed below) in horizontal mode immediately before the indentation box is placed (if there is any, i.e., if the paragraph hasn't been started with `\noindent`).

The indentation box to be typeset is available to the hook as `\IndentBox` and its automatic placement (after the hook is executed) can be prevented through `\OmitIndent`. More precisely `\OmitIndent` voids the box.

The indentation box is then typeset directly after the hook execution by something equivalent to `\box\IndentBox` followed by the current content of the token register `\everypar` that it is available to the kernel or to packages (that run some legacy code).

One has to be careful not to add any code to the hook that starts its own paragraph (e.g., by adding a `\parbox` or a `\marginpar` inside) because that would call the hook inside again (as a new paragraph is started there) and thus lead to an endless recursion ending only after exhausting the available memory. This can only be done by making sure that is not executed for the inner paragraphs (or at least not recursively forever).

**para/end** This hook is executed at the end of a paragraph when TEX is ready to return to vertical mode and after it has removed the last horizontal glue (but not any kerns) placed on the horizontal list. The code is still executed in horizontal mode so it is possible to add further horizontal material at this point, but it should not alter the mode (even a temporary exit from horizontal mode would create chaos—any attempt will cause an error message)! After the hook has ended the kernel hook `\@kernel@after@para@end` is executed and then TEX returns to vertical mode.

The hook is offered as public hook, but because of the requirement to stay within horizontal mode one needs to be careful in what is placed into the hook.<sup>20</sup>

This hook is implemented as a reversed hook.

**para/after** This hook is executed directly after TEX has returned to vertical mode and after any material that migrated out of the horizontal list (e.g., from a `\vadjust`) has processed.

---

<sup>20</sup>Maybe we should guard against that, but it would be rather tricky to implement as mode changes can happen across group boundaries so one would need to keep a private stack just for that. Well, something to ponder.

This hook should either not produce any typeset material or add only vertical material. However, for this hook starting a new paragraph is not a disaster so that it isn't prevented.

This hook is implemented as a reversed hook.

Once that hook code has been processed the kernel hook `\@kernel@after@para@after` is executed as the final action of the paragraph processing.

---

```
\@kernel@before@para@before
\@kernel@after@para@after
\@kernel@before@para@begin
\@kernel@after@para@end
```

---

As already mentioned above there are also four kernel hooks that are executed at the start and end of the processing.

`\@kernel@before@para@before` For future extensions, not currently used by the kernel.

`\@kernel@after@para@after` For future extensions, not currently used by the kernel.

`\@kernel@before@para@begin` Used by the kernel to implement tagging. This hook is executed at the very beginning of a paragraph after TeX has switched to horizontal mode but before any indentation box got added or any `\everypar` was run.

It should not generate typeset material that could alter the position. Note that it should never leave hmode, otherwise you will end with a loop! We could guard against this, but since it is an internal kernel hook that shouldn't be touched this isn't checked.

`\@kernel@after@para@end` Used by the kernel to implement tagging. It is executed directly after the public `para/end` hook. After it there is a quick check that we are still in horizontal mode, i.e., that the public hook has not mistakenly ended horizontal mode prematurely (this is an incomplete check just testing the mode and could perhaps be improved (at the cost of speed)).

## 2.2 Altered and newly provided commands

---

`\par` An explicit request for ending a paragraph is provided in plain TeX under the name `\endgraf`, which simply uses the primitive meaning (regardless of what `\par` may have as its current definition). In L<sup>A</sup>T<sub>E</sub>X `\endgraf` (with that behavior) was originally also available.

With the new paragraph handling in L<sup>A</sup>T<sub>E</sub>X, ending a paragraph means a bit more than just calling the engine's paragraph builder: the process also has to add any hook code for the end of a paragraph. Thus `\endgraf` was changed to provide this additional functionality (along with `\par` remaining subject to its current meaning).

The expl3 name for this functionality is `\para_end:`.

**Note:** *The next two commands are still under discussion and may slightly change their semantics (as described in the document) and/or their names between now and the 2021 Spring release!*

---

`\OmitIndent`  
`\para omit indent:`

Inside the `para/begin` hook one can use this command to suppress the indentation box at the start of the paragraph. (Technically it is possible to use this command outside the hook as well, but this should not be relied upon.) The box itself remains available for use.

The `expl3` name for the function is `\para omit indent:`.

---

`\IndentBox`  
`\g para indent box`

The box register holding the indentation box for the paragraph is available for inspection (or changes) inside hooks. It remains available even if the `\OmitIndent` command was used; in that case it will just not be automatically placed.

The `expl3` name for the box register is `\g para indent box`.

---

`\RawIndent`  
`\para raw indent:`  
`\RawNoindent`  
`\para raw noindent:`  
`\RawParEnd`  
`\para raw end:`

`\RawIndent hmode material \RawParEnd`  
`\RawNoindent hmode material \RawParEnd`

The commands `\RawIndent` and `\RawNoindent` are not meant for normal paragraph building (where the result is a textual paragraph in the traditional meaning of the word), but for special cases where TeX's low-level algorithm is used to achieve special effects, but where the result is not a "paragraph".

They are called "raw", because they bypass L<sup>A</sup>T<sub>E</sub>X's hook mechanism for paragraphs and simply invoke the low-level TeX algorithm. I.e., they are like the original TeX primitives `\indent` and `\noindent` (that is they execute no hooks other than `\everypar`) except that they can only be used in vertical mode and generate an error if found elsewhere.

To avoid issues a paragraph started by them should always be ended by `\RawParEnd`<sup>21</sup> and not by `\par` (or a blank line), because the latter will execute hooks which then have no counterpart at the beginning of the paragraph. It is the responsibility of the programmer to make sure that they are properly paired. This also means that one should not put arbitrary user content between these commands if that content could contain stray `\pars`.

The `expl3` names for the functions are `\para raw indent:`, `\para raw indent:` and `\para raw end:`.

## 2.3 Examples

None of the examples in this section are meant for real use as they are far too simple-minded but they should give some ideas of what could be possible if a bit more care is applied.

### 2.3.1 Testing the mechanism

The idea is to output for each paragraph encountered some information: a paragraph sequence number, a level number in roman numerals, the environment in which this paragraph appears, and the line number where the start or end of the paragraph is, e.g., something like

```
PARA: 1-i start (document env. on input line 38)
PARA: 1-i end (document env. on input line 38)
PARA: 2-ii start (minipage env. on input line 40)
PARA: 3-ii start (minipage env. on input line 40)
```

```

PARA: 3-ii end (minipage env. on input line 40)
PARA: 2-i end (document env. on input line 41)

```

As you can see paragraph 2 starts on line 40 and ends on 41 and inside a minipage started paragraph 3 (start and end on line 40). If you run this on some document you will find that L<sup>A</sup>T<sub>E</sub>X considers more things “a paragraph” than you have probably thought.

This was generated by the following hook code:

```

\newcounter{paracnt} % sequence counter
\newcounter{paralevel} % level counter

```

To support paragraph nesting we need to maintain a stack of the sequence numbers. This is most easily done using `expl3` functions, so we switch over. This is not a very general implementation, just enough for what we need and a bit of L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub>  thrown in as well. When popping, the result gets stored in `\paracntvalue` and the `\ERROR` should never happen because it means we have tried to pop from an empty stack.

```

\ExplSyntaxOn
\seq_new:N \g_para_seq
\cs_new:Npn \ParaPush
 {\seq_gpush:No \g_para_seq {\the\value{paracnt}}}
\cs_new:Npn \ParaPop {\seq_gpop:NNF \g_para_seq \paracntvalue \ERROR }
\ExplSyntaxOff

```

At the start of the paragraph increment both sequence counter and level and also save the then current sequence number on our stack.

```

\AddToHook{para/begin}{%
 \stepcounter{paracnt}\stepcounter{paralevel}%
 \ParaPush
}

```

To display the sequence number we `\typeout` the current sequence and level number. The command `\@currenvir` gives us the current environment and `\on@line` produces a space and the current input line number.

```

\typeout{PARA: \arabic{paracnt}-\roman{paralevel} start
(\@currenvir\space env.\on@line)}%

```

We also typeset the sequence number as a tiny red number in a box that takes up no horizontal space. This helps us seeing where L<sup>A</sup>T<sub>E</sub>X sees the start and end of the paragraphs in the document.

```

\llap{\color{red}\tiny\arabic{paracnt}\ }%
}

```

At the end of the paragraph we display sequence number and level again. The level counter has the correct value but we need to retrieve the right sequence value by popping it off the stack after which it is available in `\paracntvalue` the way we have set this up above.

```

\AddToHook{para/end}{%
 \ParaPop
 \typeout{PARA: \paracntvalue-\roman{paralevel} end \space\space
(\@currenvir\space env.\on@line)}%
}

```

We also typeset again a tiny red number with that value, this time sticking out to the right.<sup>22</sup> We also decrement the level counter since our level has finished.

```
\rlap{\color{red}\tiny\ \paracntvalue}%
\addtocounter{paralevel}{-1}%
}
\makeatother
```

### 2.3.2 Mark the first paragraph of each `itemize`

The code for this is rather simple. We supply some code that is executed only once inside a hook at the start of each `itemize`. We explicitly change the color back and forth so that we don't introduce grouping around the paragraph.

```
\AddToHook{env/itemize/begin}{%
\AddToHookNext{para/begin}{\color{blue}}%
\AddToHookNext{para/end}{\color{black}}%
}
```

As a result the first paragraph of each `itemize` will appear in blue.

## 2.4 Some technical notes

The code tries hard to be transparent for package code, but of course any change means that there is a potential for breaking other code. So in section we collect a few cases that may be of importance if low-level code is dealing with paragraphs that are now behaving slightly differently. The notes are from issues we observed and will probably grow over time.

### 2.4.1 Glue items between paragraphs (found with `fancypar`)

In the past L<sup>A</sup>T<sub>E</sub>X placed two glue items between two consecutive paragraphs, e.g.,

```
text1 \par text2 \par
```

would show something like

```
\glue(\parskip) 0.0 plus 1.0
\glue(\baselineskip) 5.16669
```

but now there is another `\parskip` glue (that is always 0pt):

```
\glue(\parskip) 0.0 plus 1.0
\glue(\parskip) 0.0
\glue(\baselineskip) 5.16669
```

The reason is that we generate a “fake” paragraph to gain control and safely add the early hooks, but this generates an additional glue item. That item doesn't contribute anything vertically but if somebody writes code that unravels a constructed list using `\lastbox`, `\unskip` and `\unpenalty` then the code has to remove one additional glue item or else it will fail.

---

<sup>22</sup>Note that this can alter the document pagination, because a paragraph ending in a display (e.g., an equation) will get an extra line—in that case our tiny number has an effect even though it doesn't take up any space, because it paragraph is no longer empty and thus isn't dropped!

### 3 The Implementation

```
1 <@=para>
2 (*ekernel | latexrelease)
3 \ExplSyntaxOn
4 (latexrelease) \NewModuleRelease{2021/06/01}{ltpara}
5 (latexrelease) {Paragraph-handling-and-hooks}
```

#### 3.1 Providing hooks for paragraphs

para/before  
para/after  
para/begin  
para/end

The public hooks. They are implemented as a paired set of hooks.

```
6 \hook_new_pair:nn{para/before}{para/after}
7 \hook_new_pair:nn{para/begin}{para/end}
```

(End definition for para/before and others. These functions are documented on page 295.)

\@kernel@before@para@before  
\@kernel@after@para@after  
\@kernel@before@para@begin  
\@kernel@after@para@end

The corresponding kernel hooks (for tagging and future extensions).

```
8 \let \@kernel@before@para@before \empty
9 \let \@kernel@before@para@begin \empty
10 \let \@kernel@after@para@end \empty
11 \let \@kernel@after@para@after \empty
```

(End definition for \@kernel@before@para@before and others. These functions are documented on page 296.)

\g\_para\_standard\_everypar\_tl

Whenever TeX starts a paragraph it inserts first an indentation box and then executes the tokens stored in \tex\_everypar:D (known to LATEX as \everypar). We alter this behavior slightly here, so that hooks are added into the right place. Otherwise the process change remains transparent to any legacy code for this space.

We keep the standard code to be used by \tex\_everypar:D in a separate token list because we have to switch back and forth for error recovery and so altering \tex\_everypar:D all the time should be a tiny bit faster.

```
12 \tl_new:N \g__para_standard_everypar_tl
```

Here is now its definition:

```
13 \tl_gset:Nn \g__para_standard_everypar_tl {
```

First we remove the indentation box and store it in \g\_para\_indent\_box. If there was none because the paragraph was started by \noindent the box register will be void.

```
14 \box_gset_to_last:N \g_para_indent_box
```

This will make the newly started horizontal list empty, so if we stop it now and return to vertical mode it will be dropped by TeX. We do that but inside a group so that any \parshape settings will not get lost as we need them for later.

```
15 \group_begin:
16 \tex_par:D
17 \group_end:
```

We then change \tex\_everypar:D to generate an error so that we can detect and report if the para/before hook illegally changed out of vmode.

```
18 \tex_everypar:D { \msg_error:n { hooks }{ para-mode }{before}{vertical} }
19 \@kernel@before@para@before
20 \hook_use:n {para/before}
```

Assuming the hooks have been well behaved it is time to return to horizontal mode and start the paragraph in earnest. We already have the indentation box saved away so we now have to restart the paragraph with an empty `\tex_everypar:D` and with `\tex_noindent:D`. And we need to make sure not to get another `\parskip` or rather (since we can't prevent that) that it is of zero size.

```

21 \group_begin:
22 \tex_everypar:D {}
23 \skip_zero:N \tex_parskip:D
24 \tex_noindent:D
25 \group_end:
```

That brings us back to the start of the horizontal list but we need to change `\tex_everypar:D` back to its normal content in case there are nested paragraphs coming up.

```
26 \tex_everypar:D{\g_para_standard_everypar_t1}
```

This is followed by executing the kernel and the public hook. The kernel hook is there to enable tagging.

```

27 \@kernel@before@para@begin
28 \hook_use:n {para/begin}
```

If we aren't in horizontal mode any longer the hooks above misbehaved.

```

29 \if_mode_horizontal: \else:
30 \msg_error:nnn { hooks }{ para-mode }{begin}{vertical} \fi:
```

Finally we reinsert the indentation box (unless suppressed) and then call `\everypar` the way legacy L<sup>A</sup>T<sub>E</sub>X code expects it.

However, adding the public `\everypar` is a bit tricky (see below) so we add that later, and indirectly.

```

31 __para_handle_indent:
32 % \the \everypar % <--- done differently below
33 }
```

*(End definition for `\g_para_standard_everypar_t1`.)*

`\tex_everypar:D` `\tex_everypar:D` then only has to execute `\g_para_standard_everypar_t1` by default.

```
34 \tex_everypar:D{\g_para_standard_everypar_t1}
```

*(End definition for `\tex_everypar:D`.)*

`\everypar` Tokens inserted at the beginning of the paragraph are placed into `\everypar` inside legacy L<sup>A</sup>T<sub>E</sub>X code, e.g., by the list environments or by headings to handle `\clubpenalty`, etc. Now this isn't any longer the primitive but simply a toks register used in the code above but to legacy L<sup>A</sup>T<sub>E</sub>X code that is transparent.

There is, however, a problem: a handful packages use exactly the same trick and replace the primitive with a token register and call the token register inside the renamed primitive. That is they assume that `\everypar` is the primitive and that it will still be called at the start of the paragraph even if renamed.

But if we have already replaced it by a token register then all they do is to give that token register a new name. Thus our code in `\tex_everypar:D` would call `\everypar` (which is now their token register) and the code that they added ends up in our token register which is then never used at all. A bit mind boggling I guess.

So what we have to do is not to call the token register `\everypar` by its name inside `\tex_everypar:D` but by using its actual register number.

```
35 \newtoks \everypar
```

After we have allocated a new toks register with the name `\everypar` the actual register number is available (briefly) inside `\allocationnumber`. So instead of `\the\everypar` we have to put `\the\toks<allocated number>` at the end of `\tex_everypar:D`.

So what remains doing is to append a few tokens to the token list `\g__para-standard_everypar_tl` which we do now. We use x expansion here to get the value of `\allocationnumber` in, all the other tokens should not be expanded at this point.

One important point here is to terminate the register allocation number with a real space. This space will get swallowed up when the number is read. Anything else, such as `\scan_stop:` would remain in the input and that would mean that it would interfere with `\everypar` code that attempts to scan ahead to see how the paragraph text starts.

```

36 \tl_gput_right:Nx \g__para_standard_everypar_tl {
37 \exp_not:N \the
38 \exp_not:N \toks
39 \the \allocationnumber
40 \c_space_tl
41 }
```

(End definition for `\everypar`.)

**`\g_para_indent_box`** For managing the indentation we need to provide a public accessible box register

```
42 \box_new:N \g_para_indent_box
```

(End definition for `\g_para_indent_box`. This function is documented on page 297.)

**`\__para_handle_indent:`** Adding (typesetting) the indent box is straight forward. If it was emptied before it does nothing.

```

43 \cs_new:Npn __para_handle_indent: {
44 \box_use_drop:N \g_para_indent_box
45 }
```

The declaration `\paraomitindent:` (or `\OmitIndent`) changes that to do nothing.

```

46 \cs_new:Npn \paraomitindent: {
47 \box_gclear:N \g_para_indent_box
48 }
```

(End definition for `\__para_handle_indent:..`)

**`\IndentBox`** **`\OmitIndent`** The L<sup>A</sup>T<sub>E</sub>X2<sub>ε</sub> names for the indentation box and for suppressing it for use in the para/begin hook.

```

49 \cs_set_eq:NN \IndentBox \g_para_indent_box
50 \cs_set_eq:NN \OmitIndent \paraomitindent:
```

(End definition for `\IndentBox` and `\OmitIndent`. These functions are documented on page 297.)

**`\para_end:`** Adding hooks to the end of a paragraph is similar but here we need to alter the command that is used by T<sub>E</sub>X to end horizontal mode and return to vertical mode, i.e., `\par`.

This is a bit more complicated as this command can appear anywhere either explicitly or implicitly added by T<sub>E</sub>X in certain situations:

- when using `\par` in the code or the document
- when using a blank line (which is converted to `\par`)
- when T<sub>E</sub>X finds any commands incompatible with horizontal mode it issues a `\par` and then rereads the command.

Unfortunately,  $\text{\TeX}$  has some (these days) unnecessary optimizations: if a  $\text{\vbox}$  ends and  $\text{\TeX}$  is still in horizontal mode it simply exercises the paragraph builder instead of issuing a  $\text{\par}$ . It is therefore necessary for  $\text{\LaTeX}$  to ensure that this case doesn't happen and all boxes internally have a  $\text{\par}$  command at their end.

This  $\text{\par}$  may or may not run the “par primitive” (which is always available as  $\text{\tex\_par:D}$  in  $\text{expl3}$ ); it is permissible to have a changed meaning and it is in fact changed by  $\text{\LaTeX}$  in various ways at various points inside  $\text{latex.ltx}$ . For this  $\text{\LaTeX} 2_{\varepsilon}$  code has the following conventions:  $\text{\@@par}$  and  $\text{\endgraf}$  both refer to the default meaning (in the past this was the  $\text{initex}$  primitive) while  $\text{\par}$  is the current meaning which maybe does something else.

We are now going to change this default meaning to instead run  $\text{\para_end:}$ , which ultimately executes the  $\text{initex}$  primitive but additionally adds our hooks when appropriate. This way the change is again transparent to the legacy  $\text{\LaTeX} 2_{\varepsilon}$  code.

In most cases  $\text{\para_end:}$  should behave exactly like the primitive and we achieve this by simply expanding it to the primitive which is available to us as  $\text{\tex_par:D}$ . This way we don't have to care about whether  $\text{\TeX}$  just does nothing (e.g., if in vertical mode already) or generates an error, etc.

```
51 \cs_new_protected:Npn \para_end: {
```

CCC Maybe needs more explanation. TEMP NOTE: What should happen if in outer hmode with an empty hlist?

The only case we care about is when we are in horizontal mode (i.e., doing typesetting) and not also in inner mode (i.e., making paragraphs and not building an  $\text{\hbox}$ ).

```
\bool_lazy_and:nnT
 { \mode_if_horizontal_p: }
 { \bool_not_p:n { \mode_if_inner_p: } }
 { ... }
```

Since this is executed for each and every paragraph in a document we try to stay as fast as possible, so we do not use the above construct but two conditionals instead. Using low-level  $\text{\if_mode...}$  conditions would be even faster but has the danger to conflict with conditionals in the user hooks.

If  $\text{\para_end:}$  is executed while  $\text{\TeX}$  is currently doing a low-level assignment the test for horizontal mode may get executed as part of the assignment. That is normally not an issue but we just found one case where it is:

```
\afterassignment\lst@vskip\@tempskipa \z@\par
```

If  $\text{\TeX}$  is in hmode while that assignment happens then the  $\text{\par}$  is seen in hmode because in the above case the assignment may not be finished (one should have used  $\text{\z@skip}$ ) and the  $\text{\lst@vskip}$  will get inserted into the middle of the conditional. The  $\text{\lst@vskip}$  then changes to vmode and you get a surprising error about the  $\text{para/end}$  hook having changed modes even if you don't have any hook code(!): it is the inserted  $\text{\lst@vskip}$  that is actually causing the change of mode. This is what happened when the output routines got started while a  $\text{lstlisting}$  environment (that redefines  $\text{\vskip}$  in this way) was active. This is really faulty coding, but we try to be proactive and guard the conditional so that any scanning is first stopped, thus:

```
52 \scan_stop:
53 \mode_if_horizontal:TF {
54 \mode_if_inner:F {
```

In that case the action of the primitive would be to remove the last glue (but no kerns) from the horizontal list (constructed to form a paragraph) and then to append a penalty of 10000 and the `\parfillskip`; it then passes the whole list to the paragraph builder, which breaks it into lines and `TEX` then returns to vertical mode.

What we want to do is to add this hook code at the end of the horizontal list before any of the above happens. If there was a glue item at the end of the list then it should get removed before the hook code gets added so we have to arrange for this removal.

As in other simular cases, it maybe best to add here a `\nobreak` in case the hook itself adds glue and thus creates a non-explicit and unwanted potential breakpoont. On the other hand (as has been argued) the code in the hook should perhaps have the responsibility for adding such a guard penalty in this casse. This needs further analysis and decisions (as in emails).

In either case, good documentation of these hooks is essential, covering what the hook may or should provide and all such related considerations converning the content.

There is not much point in checking if there was really a glue item at the end of the horizontal list, instead we simply try to remove one using `\tex_untskip:D`: if there wasn't one this will do nothing.

```
55 \tex_untskip:D
```

We then execute the public hook (which may add some final typeset material) followed by the kernel hook that we need for adding tagging support. None of this is supposed to change the mode—at the moment we make only a very simple test for this, more devious changes go unnoticed, but too bad as they will then probably backfire badly.

```
56 \hook_use:n{para/end}
57 \@kernel@after@para@end
58 \mode_if_horizontal:TF {
```

The final action (before getting to the point where `\tex_par:D` is called) is to add an extra glue item so that the primitive is prevented from removing intended glue (if there was some). If we don't do this and the horizontal list ends in several glue items we would end up removing two glue items instead of just the last one, which would be wrong. We use glue (rather than a kern) as that will be removed by the primitive.

There is however one other `TEX` optimization that hurts: in a sequence like this `$$ ... $$ \par` (with `\par` being the primitive) `TEX` will be in horizontal mode after the display, ready to receive further paragraph text, but since the `\par` follows immediately there is a “null” paragraph at the end and `TEX` simply throws that away. The space between `$$` and `\par` got already dropped during the display processing so the `\par` is not removing any space and appending `\parfillskip`, instead it simply goes silently to vmode.

Now if we would have added something (to prevent glue removal) that would look to `TEX` like material after the display and so we would end up with an empty paragraph just containing a penalty and `\parfillskip`.

We therefore check if the current hlist does end in glue (`\tex_lastnodedtype:D` has the value 11) and if so we add a zero-length guard skip which will be removed by the following `\tex_par:D`.

```
59 \if_int_compare:w 11 = \tex_lastnodedtype:D
60 \tex_hskip:D \c_zero_dim
61 \fi:
```

To run the `para/after` hook we first end the paragraph. This means that the `\tex_par:D` at the very end is unnecessary but executing it there unnecessarily is better than having code that tests for all the different mode possibilities.

```

62 \tex_par:D
63 \hook_use:n{para/after}
64 \kernel@after@para@after
65 }

```

If we were not horizontal mode (the F case from above) then the earlier hook `para/end` must have been at fault, so we report that.

```
66 { \msg_error:nnn { hooks }{ para-mode }{end}{horizontal} }
```

Finally close out the nested conditionals.

```

67 }
68 }

```

And then we can use the primitive to truly end the paragraph.

```

69 \tex_par:D
70 }

```

(*End definition for `\para_end`.. This function is documented on page 296.*)

`\para_raw_indent:` `\para_raw_noindent:` `\para_raw_end:`

To avoid issues a paragraph started by them should always be ended by `\para_raw_end:` and not by `\para_end:` or `\par` as the latter will execute hooks which then have no counterpart at the beginning of the paragraph. It is the responsibility of the programmer to make sure that they are properly paired.

```

71 \cs_new:Npn \para_raw_indent: {
72 \mode_if_vertical:TF
73 {
74 \tex_everypar:D {
75 \box_gset_to_last:N \g_para_indent_box
76 \tex_everypar:D { \g__para_standard_everypar_tl }
77 __para_handle_indent:
78 \the\tex_everypar }
79 }
80 { \msg_error:nn { latex2e }{ raw-para } }
81 \tex_indent:D
82 }
83 \cs_new:Npn \para_raw_noindent: {
84 \mode_if_vertical:TF
85 {
86 \tex_everypar:D {
87 \tex_everypar:D { \g__para_standard_everypar_tl }
88 \the\tex_everypar }
89 }
90 { \msg_error:nn { latex2e }{ raw-para } }
91 \tex_noindent:D
92 }
93 \cs_new_eq:NN \para_raw_end: \tex_par:D

```

(*End definition for `\para_raw_indent`:, `\para_raw_noindent`:, and `\para_raw_end`:. These functions are documented on page 297.*)

```

\RawIndent The LATEX 2 ε names for starting and ending a paragraph without adding any hooks.
\RawNoIndent 94 \cs_set_eq:NN \RawIndent \para_raw_indent:
\RawParEnd 95 \cs_set_eq:NN \RawNoindent \para_raw_noindent:
 96 \cs_set_eq:NN \RawParEnd \para_raw_end:

```

(End definition for `\RawIndent`, `\RawNoIndent`, and `\RawParEnd`. These functions are documented on page 297.)

This ends the `para` module code.

```
97 <@C=
```

```

\par Having the new default definition for \par we also have to set it up so that it gets used.
\endgraf This involves three commands: \par, \@par (to which LATEX resets \par occasionally)
\@par and \endgraf, which is another name for the “default” action of \par.

```

```

98 \cs_set_eq:NN \par \para_end:
99 \cs_set_eq:NN \@par \para_end:
100 \cs_set_eq:NN \endgraf \para_end:

```

(End definition for `\par`, `\endgraf`, and `\@par`. These functions are documented on page 296.)

While this is not integrated properly into the format we have to redo the `\everypar` setting from the kernel, otherwise that gets lost (as it happens before that file is loaded).

```
101 \everypar{\@nodocument} %% To get an error if text appears before the
```

## 3.2 The error messages

This one is used when we detect that some hook code has changed the mode where it shouldn’t, e.g., by starting or ending a paragraph. The first argument is the hook name second the mode it should have stayed in but didn’t.

```

102 \msg_new:nnnn { hooks } { para-mode }
103 {
104 Illegal~mode~ change~ in~ hook~ 'para/#1'.\\
105 Hook~ code~ did~ not~ remain~ in~ #2~ mode.
106 }
107 {
108 Paragraph~ hooks~ cannot~ change~ the~ TeX~ mode~ without~ causing~
109 endless~ recursion.~ The~ hook~ code~ in~ 'para/#1'~ needs~ to~ stay~
110 in~ #2~ mode,~ but~ it~ didn't.~ Examine~ the~ hook~
111 code~ with~ \iow_char:N \\ShowHook~ to~ find~ the~ issue.
112 }

```

And here is one used in the “raw” commands when they are used outside of vertical mode.

```

113 \msg_new:nnnn { latex2e } { raw-para }
114 {
115 Not~ in~ vertical~ mode.
116 }
117 {
118 Starting~ a~ paragraph~ with~ \iow_char:N \\RawIndent~ or~
119 \iow_char:N \\RawNoindent \\
120 (or~ \iow_char:N \\para_raw_indent:~ or~
121 \iow_char:N \\para_raw_noindent:)~ is~ only~ allowed \\
122 if~ LATEX~ is~ in~ vertical~ mode.
123 }

```

```

124 %
125 <|latexrelease>\IncludeInRelease{0000/00/00}%
126 <|latexrelease> {ltpara}{Undo-hooks-for-paragraphs}
127 <|latexrelease>
128 <|latexrelease>\let \OmitIndent \@undefined
129 <|latexrelease>\let \IndentBox \@undefined
130 <|latexrelease>\let \RawIndent \@undefined
131 <|latexrelease>\let \RawNoindent \@undefined
132 <|latexrelease>\let \RawParEnd \@undefined
133 <|latexrelease>
134 <|latexrelease>\cs_set_eq:NN \par \tex_par:D
135 <|latexrelease>\cs_set_eq:NN \@@par \tex_par:D
136 <|latexrelease>\cs_set_eq:NN \endgraf \tex_par:D
137 <|latexrelease>

```

We also need to clean up the primitive “everypar” as that should no longer execute any code by default. And, of course, make `\everypar` become the primitive again.

```

138 <|latexrelease>\tex_everypar:D {}
139 <|latexrelease>\cs_set_eq:NN \everypar \tex_everypar:D
140 <|latexrelease>
141 <|latexrelease>\EndModuleRelease
142 \ExplSyntaxOff
143 </2ekernel | latexrelease>

```

# File o ltmeta.dtx

## Abstract

This code defines the `\DocumentMetadata` interface.

## Contents

### 1 Introduction

In the past there was no dedicated location to declare settings concerning a document as a whole. Settings are placed somewhere in the preamble or with the class options or even with some package options. For some settings this can be too late, for example the pdf version can no longer be changed if a package has used code which already opened the PDF.

`\DocumentMetadata` as a new command unifies such settings in one place. It must be used before `\documentclass` but can be issued more than once there.

At the moment most of the code run by `\DocumentMetadata` is external to the format and subject to change. This includes the supported key/values.

For that reason all that happens right now in the format is to look for suitable support files and if found, to redirect the processing to them.

#### 1.1 `\DocumentMetadata`

---

`\DocumentMetadata \DocumentMetadata{\langle key-value list\rangle}`

The keys defined for `\DocumentMetadata` currently allow to set the PDF version, to set the PDF `/Lang`, to uncompress a PDF, to set the language and to declare a few PDF standards and some color profiles.

`\DocumentMetadata` is also used to activate the new PDF management code and it loads a number of required files for the PDF management code. As this forces the loading of the backend files, a backend which can't be detected automatically like `dvipdfmx`, must be set in the first `\DocumentMetadata` call (if there is more than one).

The full set of keys currently supported is documented in `documentmetadata-support.pdf` for now.

### 2 The Implementation

```
1 {*2ekernel | latexrelease}
Not needed yet but ...
2 %\ExplSyntaxOn
3 <latexrelease> \NewModuleRelease{2021/06/01}{ltmeta}
4 <latexrelease> {Document Metadata handling}
5 \protected\def\DocumentMetadata{%
6 \InputIfFileExists{documentmetadata-support.ltx}{}%
```

```
7 {}%
```

The above file is changing `\DocumentMetadata` to a suitable definition (or so we hope), so that we can try again — if not tough.

If the file can't be found we say so and carry on without it.

```
8 {%
9 \@latex@error{No support files for
10 \noexpand\DocumentMetadata found}
11 {Is the 'LaTeX-lab' bundle installed?%
12 \MessageBreak
13 Without it, the declaration is ignored.}%
14
15 \let\DocumentMetadata\@gobble
16 \DocumentMetadata
17 }%
18 %
19 <| latexrelease> \IncludeInRelease{0000/00/00}{ltmeta}%
20 <| latexrelease> {Undo Document Metadata handling}
21 <| latexrelease>
22 <| latexrelease> \let\DocumentMetadata\@undefined
23 <| latexrelease>
24 <| latexrelease> \EndModuleRelease
```

Again for the future ...

```
25 \%ExplSyntaxOff
26 </2ekernel | latexrelease>
```

Restore module prefix (if any):

```
27 <@@=
```

# File p

## ltspace.dtx

### 1 Spacing

This section deals with spacing, and line- and page-breaking.

#### 1.1 User Commands

```
\nopagebreak [⟨i⟩] : ⟨i⟩ = 0,...,4.
 Default argument = 4. Puts a penalty into the vertical list output as follows:
0 : penalty = 0
1 : penalty = \@lowpenalty
2 : penalty = \@medpenalty
3 : penalty = \@highpenalty
4 : penalty = 10000
\pagebreak [⟨i⟩] : same as except negatives of its penalty
\linebreak [⟨i⟩] : analog of the above
\nolinebreak [⟨i⟩] : analog of the above
\samepage : inhibits page breaking most places by setting the following penalties to 10000:
 \interlinepenalty
 \postdisplaypenalty
 \interdisplaylinepenalty
 \@beginparpenalty
 \@endparpenalty
 \@itempenalty
 \@secpenalty
 \interfootnotelinepenalty
\ : initially defined to be \newline
 \\[⟨length⟩] : initially defined to be \vspace{⟨length⟩}\newline
Note: * adds a \vadjust{\penalty 10000}
 OBSOLETE COMMANDS (which never made it into the manual):
 \obeycr : defines <CR> == \\relax
 \restorecr : restores <CR> to its usual meaning.
```

#### 1.2 Chris' comments

There are several aspects of the handling of space in horizontal mode that are inconsistent or do not work well in some cases. These are largely concerned with ignoring the effect of space tokens that would otherwise typeset an inter-word space.

Negating the effect of such space tokens is achieved by two mechanisms:

- `\unskip` is used to remove the glue just added by a space that has already had its effect; it is sometimes invoked after an `\ifdim` test on `\lastskip` (see below);
- `\ignorespaces` is used to ignore space-tokens yet to come.

The test done on `\lastskip` is sometimes for equality with zero and sometimes for being positive. Recall also that the test is only on the natural length of the glue and that no glue cannot be distinguished from glue whose natural length is zero: to summarise, a pretty awful test. It is not clear why these tests are not all the same; I think that they should all be for equality. One place where `\unskip` is often used is just before a `\par` (which itself internally does an `\unskip`) and one bit of code (in `\@item`) even has two `\unskips` before a `\par`. These uses may be fossil code but if they are necessary, maybe `\@killglue` would be even safer.

Such removal of glue by `\unskip` may sometimes have the wrong result, removing not the glue from a space-token but other explicit glue; this is sometimes not what is intended.

A common way to prevent such removal is to add an `\hskip\z@` after the glue that should not be removed. This protects that glue against one `\unskip` with no test but not against more than one. It does work for ‘tested `\unskips`’. This is used by `\hspace*` but not by `\hspace`; this is inconsistent as the star is supposed to prevent removal only at the beginning of a line, not at the end, or in a tabular, etc.

If this reason for removing glue were the only consideration then a tested-`\unskip` and protection by `\hskip\z@` would suffice but would need to be consistently implemented.

However, the class of invisibles, commands and environments tries to be even cleverer: one of these tries to leave only one inter-word space whenever there is one before it and one after it; and it does this quite well.

But problems can arise when there is not a space-token on both sides of it; in particular, when an invisible appears at the beginning or end of a piece of text the method still leaves one space token whereas usually in these cases it should leave none.

Also, the current rules do not work well when more than one such command appears consecutively, separated by space-tokens; it leaves glue between every other invisible.

There is also a question about what these commands should do when they occur next to spaces that do not come from space tokens but, for example, from `\hspace`. Should they still produce ‘just one space’? If so, which one? It is good to note that the manual is sufficiently cautious about invisibles that we are not obliged to make anything work.

Another interesting side-road to explore is whether the space-tokens either side of an `\hspace{...}` should be ignored.

One alternative to the current algorithm that is often suggested is that all glue around the invisible should be consolidated into a space after it (usually without stating how much glue should be put there). The command `\nolinebreak` is implemented this way (and `\linebreak` should also be). This does not work correctly for the following common case:

```
... some text
\index{some-word}
some-word and more text.
```

This is optimal coding since it is normal to index a word that gets split across a page-break on its starting page. This would, on the other hand, fix another common (and documented) failure of the current system: when the invisible is the last thing in a paragraph the space before it is not removed and, worse, it is also hidden from the paragraph-ending mechanism so that an ‘empty’ line can be created at the end of the paragraph.

Another deficiency (I think) of the current system is that the following is treated as having the `\index` command between the paragraphs, which is probably not what the author intended (since there is no empty line after it).

```
\index{beginnings}
Beginnings of paragraphs ...
```

I know of no algorithm that will handle satisfactorily even all the most common cases; note that it could be that the best algorithm may be different for different invisibles since, for example, the common uses and expected behaviour of `\index`, `\marginpar`, `\linebreak`, `\pagebreak` and `\vspace` are somewhat different. [For example, is `\vspace` ever used in the middle of a paragraph?]

One method that can (and is) used to make invisible commands produce no space when used at the beginning of text is to put in some glue that is nearly enough the same as no glue or glue of zero length in all respects except for the precise test for not being exactly equal to zero; examples of such glue are `\hskip 1sp` and, possibly better but more complex, `\hskip -1sp \hskip 1sp`. However, this only works when it is known that user-supplied text is about to start.

Some similar concerns apply to the handling of space and penalties in vertical mode; there is an extra hurdle here as `\unskip` does not work on the main vertical list. The complexity of the tests done by `\addvspace` have never been explained.

The implementation of space hacks etc for vertical mode is another major area that needs further attention; my earlier experiments did not produce much improvement over the current unsatisfactory situation.

One particular problem is what happens when the following very natural coding is used (part of the problem here is that this looks like an hmode problem, but it is not):

```
... end of text.

\begin{enumerate}
 \item \label{item:xxx} Item text.
\end{enumerate}
```

### 1.3 Some immediate actions

- Fix bug in `\linebreak`.
- Fix bug in `\*\*`.
- Reimplement `\\"`, etc, removing extra `\vadjusts` and getting better error trapping (this seems to involve a lot more tokens).
- Investigate whether `\\"`, etc need to be errors in vmode; I think that they could be noops (maybe with a warning).
- Make all(?) `\unskip`s include test for zero skip (rather than other tests or no test).
- Consider replacing `\hskip 1sp` by something better (here called an ‘infinitesimal’ skip).
- Look at all `\hskip\z@` (or similar) to see if they should be changed to an ‘infinitesimal’ skip.

- Resolve the inconsistency between `\hspace` and `\hspace*`.
- Remove unnecessary `\unskip`s.
- Investigate and rationalise the ‘newline’ code.
- Find better algorithms for all sorts of things or, easier(?), fix TeX itself.

## 1.4 The code

```

1 {*2ekernel}
2 \message{spacing,}
3 </2ekernel>
4 {*2ekernel | latexrelease}
5 <latexrelease>\IncludeInRelease{2019/10/01}%
6 <latexrelease> {\pagebreak}{Make commands robust}%

\pagebreak
\nopagebreak
7 \DeclareRobustCommand\pagebreak{\@testopt{\@no@pgbk-}4}
8 \DeclareRobustCommand\nopagebreak{\@testopt{\@no@pgbk4}{}}

(End definition for \pagebreak and \nopagebreak.)

\linebreak
\nolinebreak
9 \DeclareRobustCommand\linebreak{\@testopt{\@no@lnbk-}4}
10 \DeclareRobustCommand\nolinebreak{\@testopt{\@no@lnbk4}{}}

(End definition for \linebreak and \nolinebreak.)

\samepage
11 \DeclareRobustCommand\samepage{\interlinepenalty\@M
12 \postdisplaypenalty\@M
13 \interdisplaylinepenalty\@M
14 \beginparpenalty\@M
15 \endparpenalty\@M
16 \itempenalty\@M
17 \secpenalty\@M
18 \interfootnotelinepenalty\@M}

(End definition for \samepage.)

19 </2ekernel | latexrelease>
20 <latexrelease>\EndIncludeInRelease
21 <latexrelease>\IncludeInRelease{0000/00/00}%
22 <latexrelease> {\pagebreak}{Make commands robust}%
23 <latexrelease>
24 <latexrelease>\kernel@make@fragile\pagebreak
25 <latexrelease>\kernel@make@fragile\nopagebreak
26 <latexrelease>\kernel@make@fragile\linebreak
27 <latexrelease>\kernel@make@fragile\nolinebreak
28 <latexrelease>\kernel@make@fragile\samepage
29 <latexrelease>
30 <latexrelease>\EndIncludeInRelease
31 {*2ekernel}

```

```

\@no@pgbk
32 \def\@no@pgbk #1[#2]{%
33 \ifvmode
34 \penalty #1\@getpen{#2}%
35 \else
36 \@bsphack
37 \vadjust{\penalty #1\@getpen{#2}}%
38 \@esphack
39 \fi}

```

(End definition for \@no@pgbk.)

```

\@no@lnbk
40 \def\@no@lnbk #1[#2]{%
41 \ifvmode
42 \nolnerr
43 \else
44 \tempskipa\lastskip
45 \unskip
46 \penalty #1\@getpen{#2}%
47 \ifdim\tempskipa>\z@
48 \hskip\tempskipa
49 \ignorespaces
50 \fi
51 \fi}

```

(End definition for \@no@lnbk.)

\ The purpose of the new code is to fix a few bugs; however, it also attempts to optimize the following, in order of priority:

1. efficient execution of plain \\;
2. efficient execution of \\[...];
3. memory use;
4. name-space use.

The changes should make no difference to the typeset output. It appears to be safe to use \reserved@e and \reserved@f here (other reserved macros are somewhat disastrous).

These changes made \\newline even less robust than it had been, so now it is explicitly robust, like \\.

The internal definition of the ‘normal’ definition of \\.

```

\@normalcr
52 </2ekernel>
53 <*2ekernel | latexrelease>
54 <latexrelease> \IncludeInRelease{2020/02/02}%
55 <latexrelease> {\@normalcr}{Make robust}%
56 \protected\def\@normalcr{%
57 \let \reserved@e \relax
58 \let \reserved@f \relax
59 \@ifstar{\let \reserved@e \vadjust \let \reserved@f \nobreak
60 \xnewline}%
61 \xnewline}

```

```

62 \let\\@normalcr
63 </2ekernel | latexrelease>
64 <latexrelease>\EndIncludeInRelease
65 <latexrelease>\IncludeInRelease{0000/00/00}%
66 <latexrelease> {\@normalcr}{Make robust}%
67 <latexrelease>
68 <latexrelease>\DeclareRobustCommand\\{%
69 <latexrelease> \let \reserved@e \relax
70 <latexrelease> \let \reserved@f \relax
71 <latexrelease> \@ifstar{\let \reserved@e \vadjust \let \reserved@f \nobreak
72 <latexrelease> \xnewline}%
73 <latexrelease> \xnewline}
74 <latexrelease>\expandafter\let\expandafter\@normalcr
75 <latexrelease> \csname\expandafter\gobble\string\\ \endcsname
76 <latexrelease>
77 <latexrelease>\EndIncludeInRelease
78 <*2ekernel>

```

(*End definition for \\ and \@normalcr.*)

**\@vspace@calcify** Helper command to produce a \vskip that is first run through \setlength. This way the calc package can operate on the argument value.

```

79 </2ekernel>
80 <*2ekernel | latexrelease>
81 <latexrelease>\IncludeInRelease{2020/10/01}%
82 <latexrelease> {\@vspace@calcify}{Add calc support}%
83 \def\@vspace@calcify#1{\begingroup\setlength\skip@{#1}\vskip\skip@\endgroup}
84 </2ekernel | latexrelease>
85 <latexrelease>\EndIncludeInRelease
86 <latexrelease>\IncludeInRelease{0000/00/00}%
87 <latexrelease> {\@vspace@calcify}{Add calc support}%
88 <latexrelease>
89 <latexrelease>\let\@vspace@calcify\@undefined
90 <latexrelease>\EndIncludeInRelease
91 <*2ekernel>

```

(*End definition for \@vspace@calcify.*)

**\newline** A simple form of the ‘normal’ definition of \\.

```

92 \DeclareRobustCommand\newline{\@normalcr\relax}

```

(*End definition for \newline.*)

**\@xnewline**

```

93 \def\@xnewline{\@ifnextchar[%] bracket matching
94 \@newline
95 {\@gnewline\relax}}

```

(*End definition for \@xnewline.*)

**\@newline**

```

96 </2ekernel>
97 <*2ekernel | latexrelease>
98 <latexrelease>\IncludeInRelease{2020/10/01}%

```

```

99 <{latexrelease} {\@newline}{\newline calc support}%
100 \def\@newline[#1]{\let \reserved@e \vadjust
101 \gnewline {\@vspace@calcify{#1}}}
102 </2ekernel | latexrelease>
103 <{latexrelease}\EndIncludeInRelease
104 <{latexrelease}\IncludeInRelease{0000/00/00}%
105 <{latexrelease} {\@newline}{\newline calc support}%
106 <{latexrelease}>
107 <{latexrelease}\def\@newline[#1]{\let \reserved@e \vadjust
108 <{latexrelease} \gnewline {\vskip #1}}
109 <{latexrelease}\EndIncludeInRelease
110 <{*2ekernel}>

```

(End definition for `\@newline`.)

`\@gnewline` The `\nobreak` added to prevent null lines when `\`\\` ends an overfull line. Change made 24 May 89 as suggested by Frank Mittelbach and Rainer Schöpf

```

111 \def\@gnewline #1{%
112 \ifvmode
113 \nolnerr
114 \else
115 \unskip \reserved@e {\reserved@f#1}\nobreak \hfil \break
116 \fi}

```

(End definition for `\@gnewline`.)

`\@getpen`

```

117 \def\@getpen#1{\ifcase #1 \z@ \or \clowpenalty\or
118 \medpenalty \or \chighpenalty
119 \else \z@\fi}

```

(End definition for `\@getpen`.)

`\if@nobreak` Switch used to avoid page breaks caused by `\label` after a section heading, etc. It should be **GLOBALLY** set true after the `\nobreak` and **globally** set false by the next invocation of `\everypar`.

Commands that reset `\everypar` should globally set it false if appropriate.

```

120 \def\@nobreakfalse{\global\let\if@nobreak\iffalse}
121 \def\@nobreaktrue {\global\let\if@nobreak\iftrue}
122 \z@\nobreakfalse

```

(End definition for `\if@nobreak`.)

`\@savsk` Registers used to save the space factor and last skip.

```

123 \newdimen\@savsk
124 \newcount\@savsf

```

(End definition for `\@savsk` and `\@savsf`.)

\@bsphack \@bsphack and \@esphack used by macros such as \index and \begin{@float} ... \end{@float} that want to be invisible — i.e., not leave any extra space when used in the middle of text. Such a macro should begin with \@bsphack and end with \@esphack. The macro in question should not create any text, nor change the mode.

Before giving the current definition we give an extended definition that is currently not used (because it doesn't work as advertised:-)

These are generalised hacks which attempt to do sensible things when ‘invisible commands’ appear in vmode too.

They need to cope with space in both hmode (plus spacefactor) and vmode, and also cope with breaks etc. In vmode this means ensuring that any following \addvspace, etc sees the correct glue in \lastskip.

In fact, these improved versions should be used for other cases of ‘whatsits, thingies etc’ which should be invisible. They are only for commands, not environments (see notes on \@EspHack).

BTW, anyone know why the standard hacks are surrounded by \ifmmode\else rather than simply \ifhmode?

And are there any cases where saving the spacefactor is essential? I have some extensions where it is, but it does not appear to be so in the standard uses.

```
def \@bsphack{%
 \relax \ifvmode
 \@savsk \lastskip
 \ifdim \lastskip=\z@
 \else
 \vskip -\lastskip
 \fi
\else
 \ifhmode
 \@savsk \lastskip
 \@savsf \spacefactor
 \fi
\fi
```

I think that, in vmode, it is the safest to put in a \nobreak immediately after such things since writes, inserts etc followed by glue give valid breakpoints and, in general, it is possible to create breaks but impossible to destroy them.

```
def \@esphack{%
 \relax \ifvmode
 \nobreak
 \ifdim \@savsk=\z@
 \else
 \vskip\@savsk
 \fi
\else
 \ifhmode
 \spacefactor \@savsf
 \ifdim \@savsk>\z@
 \ignorespaces
 \fi
 \fi
\fi
```

```

\fi
\fi
```

For the moment we are going to ignore the vertical versions until they are correct.

```

125 \def\@bsphack{%
126 \relax
127 \ifhmode
128 \csavsk\lastskip
129 \csavsf\spacefactor
130 \fi}
```

(End definition for `\@bsphack`.)

- `\@esphack` Companion to `\@bsphack`. If this command is not properly paired with `\@bsphack` one might end up with a low-level TeX error: “BAD spacefactor”. One possible cause is calling `\@bsphack` in vertical mode, then doing something that gets you (sometimes) into horizontal mode and finally calling `\@esphack`. Even if no error is generated that is wrong, because `\@esphack` will then use the saved values for `\@savsk` and `\@savsf` from some earlier invocation of `\@bsphack` which will have nothing to do with the current situation.

```

131 </2ekernel>
132 <texrelease>\IncludeInRelease{2018/10/10}%
133 <texrelease> {\@esphack}{hyphenation and nobreak after space hack}%
134 <2ekernel | texrelease>
135 \def\@esphack{%
136 \relax
137 \ifhmode
138 \spacefactor\@savsf
139 \ifdim\@savsk>\z@
140 \ifdim\lastskip=\z@
141 \nobreak \hskip\z@skip
142 \fi
143 \ignorespaces
144 \fi
145 \else
146 \ifvmode
147 \if\nobreak\nobreak\else\if\noskipsec\nobreak\fi\fi
148 \fi
149 \fi}%
150 <2ekernel | texrelease>
151 <texrelease>\EndIncludeInRelease
152 <texrelease>\IncludeInRelease{2015/10/01}%
153 <texrelease> {\@esphack}{hyphenation and nobreak after space hack}%
154 <texrelease>\def\@esphack{%
155 <texrelease> \relax
156 <texrelease> \ifhmode
157 <texrelease> \spacefactor\@savsf
158 <texrelease> \ifdim\@savsk>\z@
159 <texrelease> \ifdim\lastskip=\z@
160 <texrelease> \nobreak \hskip\z@skip
161 <texrelease> \fi
```

```

162 〈\latexrelease〉 \ignorespaces
163 〈\latexrelease〉 \fi
164 〈\latexrelease〉 \fi}%
165 〈\latexrelease〉\EndIncludeInRelease
166 〈\latexrelease〉\IncludeInRelease{2015/01/01}%
167 〈\latexrelease〉 {\@esphack}{hyphenation and nobreak after space hack}%
168 〈\latexrelease〉\def\@esphack{%
169 〈\latexrelease〉 \relax
170 〈\latexrelease〉 \ifhmode
171 〈\latexrelease〉 \spacefactor\@savsf
172 〈\latexrelease〉 \ifdim\@savsk>\z@
173 〈\latexrelease〉 \nobreak \hskip\z@skip
174 〈\latexrelease〉 \ignorespaces
175 〈\latexrelease〉 \fi
176 〈\latexrelease〉 \fi}%
177 〈\latexrelease〉\EndIncludeInRelease
178 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
179 〈\latexrelease〉 {\@esphack}{hyphenation and nobreak after space hack}%
180 〈\latexrelease〉\def\@esphack{%
181 〈\latexrelease〉 \relax
182 〈\latexrelease〉 \ifhmode
183 〈\latexrelease〉 \spacefactor\@savsf
184 〈\latexrelease〉 \ifdim\@savsk>\z@
185 〈\latexrelease〉 \ignorespaces
186 〈\latexrelease〉 \fi
187 〈\latexrelease〉 \fi}%
188 〈\latexrelease〉\EndIncludeInRelease
189 〈*2ekernel〉

```

(End definition for \@esphack.)

\@Eshack A variant of \@esphack that sets the @ignore switch to true (as \@esphack used to do previously). This is currently used only for floats and similar environments. w

```

190 〈/2ekernel〉
191 〈\latexrelease〉\IncludeInRelease{2015/01/01}%
192 〈\latexrelease〉 {\@Eshack}{hyphenation after space hack}%
193 〈*2ekernel | \latexrelease〉
194 \def\@Eshack{%
195 \relax
196 \ifhmode
197 \spacefactor\@savsf
198 \ifdim\@savsk>\z@
199 \nobreak \hskip\z@skip
200 \@ignoretrue
201 \ignorespaces
202 \fi
203 \fi}%
204 〈/2ekernel | \latexrelease〉
205 〈\latexrelease〉\EndIncludeInRelease
206 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
207 〈\latexrelease〉 {\@Eshack}{hyphenation after space hack}%
208 〈\latexrelease〉\def\@Eshack{%
209 〈\latexrelease〉 \relax
210 〈\latexrelease〉 \ifhmode

```

```

211 〈\latexrelease〉 \spacefactor\@savsf
212 〈\latexrelease〉 \ifdim\@savsk>\z@
213 〈\latexrelease〉 \ignorespacestrue
214 〈\latexrelease〉 \ignorespaces
215 〈\latexrelease〉 \fi
216 〈\latexrelease〉 \fi}%
217 〈\latexrelease〉\EndIncludeInRelease
218 {*2ekernel}

```

(End definition for \@EspHack.)

\@vbsphack Another variant which is useful for invisible things which should not live in vmode (this is how some people feel about marginals).

If it occurs in vmode then it enters hmode and ensures that \@savsk is nonzero so that the \ignorespaces is put in later. It is not used at present.

```

\def \@vbsphack{ %
 \relax \ifvmode
 \leavevmode
 \@savsk 1sp
 \@savsf \spacefactor
 \else
 \ifhmode
 \@savsk \lastskip
 \@savsf \spacefactor
 \fi
 \fi
}

```

(End definition for \@vbsphack.)

## 1.5 Vertical spacing

L<sup>A</sup>T<sub>E</sub>X supports the plain T<sub>E</sub>X commands \smallskip, \medskip and \bigskip. However, it redefines them using \vspace instead of \skip.

Extra vertical space is added by the command \addvspace{\<skip>}, which adds a vertical skip of <skip> to the document. The sequence \addvspace{\<s1>} \addvspace{\<s2>} is equivalent to \addvspace{\<maximum of s1, s2>}.

\addvspace should be used only in vertical mode, and gives an error if it's not. The \addvspace command does *not* add vertical space if @minipage is true. The minipage environment uses this to inhibit the addition of extra vertical space at the beginning.

Penalties are put into the vertical list with the \addpenalty{\<penalty>} command. It works properly when \addpenalty and \addvspace commands are mixed.

The @nobreak switch is set true used when in vertical mode and no page break should occur. (Right now, it is used only by the section heading commands to inhibit page breaking after a heading.)

```

\addvspace{SKIP} ==
BEGIN
 if vmode
 then if @minipage

```

```

 else if \lastskip =0
 then \vskip SKIP
 else if \lastskip < SKIP
 then \vskip -\lastskip
 \vskip SKIP
 else if SKIP < 0 and \lastskip >= 0
 then \vskip -\lastskip
 \vskip \lastskip + SKIP
 fi fi fi fi
 else useful error message (CAR).
fi
END

```

\@xaddvskip Internal macro for \vspace handling the case that space has previously been added.

```

219 \def\@xaddvskip{%
220 \ifdim\lastskip<\@tempskipb
221 \vskip-\lastskip
222 \vskip\@tempskipb
223 \else
224 \ifdim\@tempskipb<\z@
225 \ifdim\lastskip<\z@
226 \else
227 \advance\@tempskipb\lastskip
228 \vskip-\lastskip
229 \vskip \@tempskipb
230 \fi
231 \fi
232 \fi}

```

(End definition for \@xaddvskip.)

\addvspace Add vertical space taking into account space already added, as described above.

```

233 </2ekernel>
234 <*2ekernel | latexrelease>
235 <latexrelease>\IncludeInRelease{2020/10/01}%
236 <latexrelease> {\addvspace}{\addvspace calc support}%
237 \def\addvspace#1{%
238 \ifvmode
239 \if@minipage\else
240 \ifdim \lastskip =\z@
241 \vspace@calcify{#1}%
242 \else
243 \setlength\@tempskipb{#1}%
244 \vskip\@xaddvskip
245 \fi
246 \fi
247 \else
248 \noitemerr
249 \fi}
250 </2ekernel | latexrelease>
251 <latexrelease>\EndIncludeInRelease
252 <latexrelease>\IncludeInRelease{0000/00/00}%
253 <latexrelease> {\addvspace}{\addvspace calc support}%

```

```

254 〈\latexrelease〉
255 〈\latexrelease〉\def\addvspace#1{%
256 〈\latexrelease〉 \ifvmode
257 〈\latexrelease〉 \if@minipage\else
258 〈\latexrelease〉 \ifdim \lastskip =\z@
259 〈\latexrelease〉 \vskip #1\relax
260 〈\latexrelease〉 \else
261 〈\latexrelease〉 \tempskipb#1\relax
262 〈\latexrelease〉 \xaddvskip
263 〈\latexrelease〉 \fi
264 〈\latexrelease〉 \fi
265 〈\latexrelease〉 \else
266 〈\latexrelease〉 \noitemerr
267 〈\latexrelease〉 \fi}
268 〈\latexrelease〉\EndIncludeInRelease
269 〈*2ekernel〉

```

(End definition for `\addvspace`.)

### \addpenalty

```

270 〈/2ekernel〉
271 〈\latexrelease〉\IncludeInRelease{2015/01/01}%
272 〈\latexrelease〉 {\addpenalty}{\addpenalty}%
273 〈*2ekernel | latexrelease〉

```

Fix provided by Donald (though the original fix was not good enough). In 2005 Plamen Tanovski discovered that this fix wasn't good enough either as the `\vskip` kept getting bigger if several `\addpenalty` commands followed each other. Donald kindly send a new fix.

```

274 \def\addpenalty#1{%
275 \ifvmode
276 \if@minipage
277 \else
278 \if@nobreak
279 \else
280 \ifdim\lastskip=\z@
281 \penalty#1\relax
282 \else
283 \tempskipb\lastskip

```

We have to make sure the final `\vskip` seen by TeX is the correct one, namely `\tempskipb`. However we may have to adjust for `\prevdepth` when placing the penalty but that should not affect the skip we pass on to TeX.

```

284 \begingroup
285 \tempskipa\tempskipb
286 \advance \tempskipb
287 \ifdim\prevdepth>\maxdepth\maxdepth\else

```

If `\prevdepth` is -1000pt due to `\nointerlineskip` we better not add it!

```

288 \ifdim \prevdepth = -\@m\p@ \z@ \else \prevdepth \fi
289 \fi
290 \vskip -\tempskipb
291 \penalty#1%
292 \ifdim\tempskipa=\tempskipb

```

Do nothing if the `\prevdepth` check made no adjustment.

293                  `\else`

Combine the prevdepth adjustment into a single skip.

294                  `\advance\@tempskipb -\@tempskipa`  
295                  `\vskip \@tempskipb`  
296                  `\fi`

The final skip is always the specified length.

297                  `\vskip \@tempskipa`  
298                  `\endgroup`  
299                  `\fi`  
300                  `\fi`  
301                  `\fi`  
302                  `\else`  
303                  `\@noitemerr`  
304                  `\fi} %`  
  
305 `//2ekernel | latexrelease)`  
306 `\langle latexrelease\rangle\EndIncludeInRelease`  
307 `\langle latexrelease\rangle\IncludeInRelease{0000/00/00}%`  
308 `\langle latexrelease\rangle \{\addpenalty\}{\addpenalty}%`  
309 `\langle latexrelease\rangle\def\addpenalty#1{%`  
310 `\langle latexrelease\rangle \ifvmode`  
311 `\langle latexrelease\rangle \if@minipage`  
312 `\langle latexrelease\rangle \else`  
313 `\langle latexrelease\rangle \if@nobreak`  
314 `\langle latexrelease\rangle \else`  
315 `\langle latexrelease\rangle \ifdim\lastskip=\z@`  
316 `\langle latexrelease\rangle \penalty#1\relax`  
317 `\langle latexrelease\rangle \else`  
318 `\langle latexrelease\rangle \@tempskipb\lastskip`  
319 `\langle latexrelease\rangle \vskip -\lastskip`  
320 `\langle latexrelease\rangle \penalty#1%`  
321 `\langle latexrelease\rangle \vskip\@tempskipb`  
322 `\langle latexrelease\rangle \fi`  
323 `\langle latexrelease\rangle \fi`  
324 `\langle latexrelease\rangle \fi`  
325 `\langle latexrelease\rangle \else`  
326 `\langle latexrelease\rangle \@noitemerr`  
327 `\langle latexrelease\rangle \fi} %`  
328 `\langle latexrelease\rangle\EndIncludeInRelease`  
329 `*2ekernel}`

(End definition for `\addpenalty`.)

`\vspace` The new code for these commands depends on the following facts:  
`\@vspace`  
`\@vspacer`

- The value of `prevdepth` is changed only when a box or rule is created and added to a vertical list;
- The value of `prevdepth` is used only when a box is created and added to a vertical list;
- The value of `prevdepth` is always local to the building of one vertical list.

330 `\DeclareRobustCommand\vspace{\@ifstar\@vspacer\@vspace}`

```

331 </2ekernel>
332 <*2ekernel | latexrelease>
333 <latexrelease>\IncludeInRelease{2020/10/01}%
334 <latexrelease> {\@vspace}{Support calc in \vspace}%

```

We support calc syntax in the argument and therefore use `\setlength`.

```

335 \def\@vspace #1{%
336 \ifvmode
337 \@vspace@calcify{#1}%
338 \vskip\z@skip
339 \else
340 \@bsphack
341 \vadjust{\@restorepar
342 \@vspace@calcify{#1}%
343 \vskip\z@skip
344 }%
345 \@esphack
346 \fi}
347 \def\@vspacer#1{%
348 \ifvmode
349 \dimen@\prevdepth
350 \hrule\@height\z@
351 \nobreak
352 \@vspace@calcify{#1}%
353 \vskip\z@skip
354 \prevdepth\dimen@
355 \else
356 \@bsphack
357 \vadjust{\@restorepar
358 \hrule\@height\z@
359 \nobreak
360 \@vspace@calcify{#1}%
361 \vskip\z@skip}%
362 \@esphack
363 \fi}
364 </2ekernel | latexrelease>
365 <latexrelease>\EndIncludeInRelease
366 <latexrelease>\IncludeInRelease{0000/00/00}%
367 <latexrelease> {\@vspace}{Support calc in \vspace}%
368 <latexrelease>
369 <latexrelease>\def\@vspace #1{%
370 <latexrelease> \ifvmode
371 <latexrelease> \vskip #1
372 <latexrelease> \vskip\z@skip
373 <latexrelease> \else
374 <latexrelease> \@bsphack
375 <latexrelease> \vadjust{\@restorepar
376 <latexrelease> \vskip #1
377 <latexrelease> \vskip\z@skip
378 <latexrelease> }%
379 <latexrelease> \@esphack
380 <latexrelease> \fi}
381 <latexrelease>\def\@vspacer#1{%
382 <latexrelease> \ifvmode

```

```

383 〈\latexrelease〉 \dimen@\prevdepth
384 〈\latexrelease〉 \hrule \@height\z@
385 〈\latexrelease〉 \nobreak
386 〈\latexrelease〉 \vskip #1
387 〈\latexrelease〉 \vskip\z@skip
388 〈\latexrelease〉 \prevdepth\dimen@
389 〈\latexrelease〉 \else
390 〈\latexrelease〉 \@bsphack
391 〈\latexrelease〉 \vadjust{\@restorepar
392 〈\latexrelease〉 \hrule \@height\z@
393 〈\latexrelease〉 \nobreak
394 〈\latexrelease〉 \vskip #1
395 〈\latexrelease〉 \vskip\z@skip}%
396 〈\latexrelease〉 \@esphack
397 〈\latexrelease〉 \fi}
398 〈\latexrelease〉\EndIncludeInRelease
399 〈*2ekernel〉

```

(End definition for `\vspace`, `\@vspace`, and `\@vspace@`.)

```

\smallskip
\medskip 400 \def\smallskip{\vspace\smallskipamount}
\bigskip 401 \def\medskip{\vspace\medskipamount}
402 \def\bigskip{\vspace\bigskipamount}

```

(End definition for `\smallskip`, `\medskip`, and `\bigskip`.)

```

\smallskipamount
\medskipamount 403 \newskip\smallskipamount \smallskipamount=3pt plus 1pt minus 1pt
\bigskipamount 404 \newskip\medskipamount \medskipamount =6pt plus 2pt minus 2pt
405 \newskip\bigskipamount \bigskipamount =12pt plus 4pt minus 4pt

```

(End definition for `\smallskipamount`, `\medskipamount`, and `\bigskipamount`.)

## 1.6 Horizontal space (and breaks)

`\nobreakdashes` This idea is borrowed from the `amsmath` package but here we define a robust command.

This command is a low-level command designed for use only before hyphens or dashes (such as `-`, `--`, or `---`).

It could probably be better implemented: it may need its own private token register and temporary command.

Setting the hyphen in a box and then unboxing it means that the normal penalty will not be added after it—and if the penalty is not there a break will not be taken (unless an explicit penalty or glue follows, thus the final `\nobreak`).

Note that even if it is not followed by a `'`, it still leaves vmode and sets the space-factor; so use it carefully!

```

406 \DeclareRobustCommand{\nobreakdashes}{%
407 \leavevmode
408 \toks@{}%
409 \def\reserved@a##1{\toks@\expandafter{\the\toks@-}%
410 \futurelet\@let@token \reserved@b}%
411 \def\reserved@b {\ifx\@let@token -%
412 \expandafter\reserved@a

```

```

413 \else
414 \setbox\z@ \hbox{\the\toks@\nobreak}%
415 \unhbox\z@
416 \spacefactor\sfcodes`-
417 \fi}%
418 \futurelet\@let@token \reserved@b
419 }

```

(End definition for `\nobreakdashes`.)

`\nobreakspace  
  \@xobeysp`

This is a robust command that produces a horizontal space at which, in paragraph-mode, a line-break is not possible. We then define an active `\~` to expand to it since this is the documented behaviour of `\~`. One reason for introducing this is that some 8-bit input encodings have a slot for such a space and we do not want to use active characters as the L<sup>A</sup>T<sub>E</sub>X internal commands.

The braces in the definition of `\~` are needed to ensure that a following space is preserved when reading to/from internal files.

We need to keep `\@xobeysp` as it is widely used; so here it is let to the non-robust command `\nobreakspace`.

```

420 \DeclareRobustCommand{\nobreakspace}{%
421 \leavevmode\nobreak\ }
422 \catcode`\~=13
423 \def~{\nobreakspace{}}
424 \expandafter\let\expandafter\@xobeysp\csname nobreakspace \endcsname

```

(End definition for `\nobreakspace` and `\@xobeysp`.)

- Placed before a `\.`, makes it a sentence-ending period. Does the right thing for other punctuation marks as well. Does this by setting spacefactor to 1000.

```

425 </2ekernel>
426 <latexrelease>\IncludeInRelease{2015/01/01}%
427 <latexrelease> {\@{Space after \@{}}%
428 <*2ekernel | latexrelease>
429 \def\@{\spacefactor\@m{}}%
430 </2ekernel | latexrelease>
431 <latexrelease>\EndIncludeInRelease
432 <latexrelease>\IncludeInRelease{0000/00/00}%
433 <latexrelease> {\@{Space after \@{}}%
434 <latexrelease>\def\@{\spacefactor\@m{}}%
435 <latexrelease>\EndIncludeInRelease
436 <*2ekernel>

```

(End definition for `\@{}`.)

`\hspace`

```
437 \DeclareRobustCommand\hspace{\@ifstar\@hspace\@hspace}
```

(End definition for `\hspace`.)

```

\@hspace
 438 </2ekernel>
 439 <*2ekernel | latexrelease>
 440 <latexrelease>\IncludeInRelease{2020/10/01}%
 441 <latexrelease> {\@hskip}{Support calc with \hskip}%
 442 \def\@hskip#1{\begin{group}\setlength\skip@{\#1}\hskip\skip@\endgroup}
 443 </2ekernel | latexrelease>
 444 <latexrelease>\EndIncludeInRelease
 445 <latexrelease>\IncludeInRelease{0000/00/00}%
 446 <latexrelease> {\@hskip}{Support calc with \hskip}%
 447
 448 <latexrelease>
 449 <latexrelease>\def\@hskip#1{\hskip #1\relax}
 450 <latexrelease>\EndIncludeInRelease
 451 <*2ekernel>

(End definition for \@hskip.)
```

\@hspacer Extra \hskip Opt added 1985/17/12 to guard against a following \unskip \relax added 13 Oct 88 for usual TeX lossage replaced both changes by \hskip\z@skip 27 Nov 91

```

 452 \def\@hspacer#1{\vrule \width\z@\nobreak
 453 \hskip\#1\z@skip}
```

(End definition for \@hspacer.)

```

\fill
 454 \newskip\fill
 455 \fill = Opt plus 1fill

(End definition for \fill.)
```

\stretch

```

 456 \def\stretch#1{\z@ \oplus #1fill\relax}
```

(End definition for \stretch.)

```

 457 </2ekernel>
 458 <*2ekernel | latexrelease>
 459 <latexrelease>\IncludeInRelease{2018/12/01}%
 460 <latexrelease> {\thinspace}{Start LR-mode}%
```

\enspace

```

 461 \DeclareRobustCommand\enspace{\leavevmode@ifvmode\kern.5em }
```

(End definition for \enspace.)

\leavevmode@ifvmode Leave vmode but only if we are really in vmode, otherwise the expansion is empty (which is not the case with the default definition).

```

 462 \protected\def\leavevmode@ifvmode{\ifvmode\expandafter\indent\fi}
```

```

(End definition for \leavevmode@ifvmode.)

463 </2ekernel | latexrelease>
464 <latexrelease>\EndIncludeInRelease
465 <latexrelease>\IncludeInRelease{0000/00/00}%
466 <latexrelease> {\thinspace}{Start LR-mode}%
467 <latexrelease>\def\thinspace{\kern .16667em }
468 <latexrelease>\def\negthinspace{\kern-.16667em }
469 <latexrelease>\def\enspace{\kern.5em }
470 <latexrelease>\let\leavevmode@ifvmode@undefined
471 <latexrelease>\EndIncludeInRelease
472 <*2ekernel>
```

```

\enskip
\quad 473 \def\enskip{\hskip.5em\relax}
\quad 474 \def\quad{\hskip1em\relax}
\quad 475 \def\quad{\hskip2em\relax}
```

(End definition for \enskip, \quad, and \quad.)

For Unicode engines, make the Unicode soft hyphen an active character defined as \-.

```

476 \ifx\Umathcode\@undefined\else
477 \catcode "AD=13
478 \def^^ad{\-}
479 \fi
```

\obeycr The following definitions will probably get deleted or moved to compatibility mode soon.  
\restorecr
480 {\catcode`^^M=13 \gdef\obeycr{\catcode`^^M13 \def^^M{\relax}%
481 @gobblecr}%
482 {\catcode`^^M=13 \gdef@gobblecr{@ifnextchar
483 @gobble\ignorespaces}%
484 \gdef\restorecr{\catcode`^^M5 }}

(End definition for \obeycr and \restorecr.)

```
485 </2ekernel>
```

# File q

## ltlogos.dtx

### 1 Logos

Various logos are defined here.

- \TeX The \TeX logo, adjusted so that a full stop after the logo counts as ending a sentence.

```
1 <*2ekernel>
2 \DeclareRobustCommand{\TeX}{\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX\@}
```

(End definition for \TeX.)

- \LaTeX The \LaTeX logo.

```
3 \DeclareRobustCommand{\LaTeX}{\kern-.36em%
4 {\sbox{z@T}%
5 \vbox to\ht{z@}{\hbox{\check@mathfonts%
6 \fontsize\sf@size\z@%
7 \math@fontsfalse\selectfont%
8 A}%
9 \vss}%
10 }%
11 \kern-.15em%
12 \TeX}
```

(End definition for \LaTeX.)

- \LaTeXe The \LaTeX<sub>2ε</sub> logo as proposed by A-W designers.

```
13 \DeclareRobustCommand{\LaTeXe}{\mbox{\m@th%
14 \if b\expandafter\@car\f@series\@nil\boldmath\fi%
15 \LaTeX\kern.15em\textstyle\varepsilon\}}%
16 </2ekernel>
```

(End definition for \LaTeXe.)

# File r

## ltfiles.dtx

### 1 File Handling

The following user commands are defined in this part:

|                    |                                                                                                                                                                                             |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \document          | (ie \begin{document})                                                                                                                                                                       |
|                    | Reads in the .AUX files and \catcode's @ to 12.                                                                                                                                             |
| \nofiles           | Suppresses all file output by setting \@filesw false.                                                                                                                                       |
| \includeonly       | \{(NAME1, ... ,NAMEn)\}                                                                                                                                                                     |
|                    | Causes only parts NAME1, ... ,NAMEn to be read by their \include commands. Works by setting partsw true and setting \@partlist to NAME1, ... ,NAMEn.                                        |
| \include           | \{(NAME)\}                                                                                                                                                                                  |
|                    | Does an \input NAME unless \@partsw is true and NAME is not in \@partlist. If \@filesw is true, then it directs .AUX output to NAME.AUX, including a checkpoint at the end.                 |
| \input             | \{(NAME)\}                                                                                                                                                                                  |
|                    | The same as TeX's \input, except it allows optional braces around the file name. In LATEX 2 <sub>E</sub> , it also avoids the primitive 'missing file' error, if the file can not be found. |
| \IfFileExists      | \{(NAME)\}\{(then)\}\{(else)\}                                                                                                                                                              |
|                    | If the file exists on the system, execute then otherwise execute else.                                                                                                                      |
| \InputIfFileExists | \{(NAME)\}\{(then)\}\{(else)\}                                                                                                                                                              |
|                    | If the file exists on the system, execute then and input NAME otherwise execute else.                                                                                                       |

*Historical LATEX 2.09 comments (not necessarily accurate any more):*

1 <\*2ekernel>  
2 \message{files,}

#### VARIABLES, SWITCHES AND INTERNAL COMMANDS:

|             |                                                                                                 |
|-------------|-------------------------------------------------------------------------------------------------|
| @mainaux    | : Output file number for main .AUX file.                                                        |
| @partaux    | : Output file number for current part's .AUX file.                                              |
| @auxout     | : Either @mainout or @partout, depending on which .AUX file output goes to.                     |
| @input{foo} | : If file foo exists, then \input's it, otherwise types a warning message.                      |
| @filesw     | : Switch – set false if no .AUX, .TOC, .IDX etc files are to be written                         |
| @partsw     | : Set true by a \includeonly command.                                                           |
| @partlist   | : Set to the argument of the \includeonly command.                                              |
| \cp@FOO     | : The checkpoint for \include'd file FOO.TEX, written by \@writeckpt at the end of file FOO.AUX |

\includeonly{FILELIST} ==  
BEGIN

```

\@partsw := T
\@partlist := FILELIST
END

\include{FILE} ==
BEGIN
 \clearpage
 if \@files w = T
 then \immediate\write\@mainaux{\string\@input{FILE.AUX}}
 fi
 if \@partsw = T
 then \@tempswa := F
 \reserved@b == FILE
 for \reserved@a := \@partlist
 do if eval(\reserved@a) = eval(\reserved@b)
 then \@tempswa := T fi
 od
 fi

 if \@tempswa = T
 then \@auxout := \@partaux
 if \@files w = T
 then \immediate\openout\@partaux{FILE.AUX}
 \immediate\write\@partaux{\relax}
 fi
 \@input{FILE.TEX}
 \clearpage
 \@writeckpt{FILE}
 if @files w then \closeout\@partaux fi
 \@auxout := \@mainaux
 else \cp@FILE
 fi
END

\@writeckpt{FILE} ==
BEGIN
 if \@files w = T
 \immediate\write on file \@partaux:
 \@setckpt{FILE}{% }
 for \reserved@a := \cl@ckpt
 do \immediate\write on file \@partaux:
 \global\string\setcounter
 {eval(\reserved@a)}{eval(\c@eval(\reserved@a))}
 od
 \immediate\write on file \@partaux: %
 fi
END

\@setckpt{FILE}{LIST} ==
BEGIN

```

```
G \cp@FILE := LIST
END
```

```
INITIALIZATION
\@tempswa := T
```

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

```
\@mainaux
\@partaux 3 \newwrite\@mainaux
4 \newwrite\@partaux
```

*(End definition for \@mainaux and \@partaux.)*

```
\if@filesw
\if@partsw 5 \newif\if@filesw \@fileswtrue
6 \newif\if@partsw \@partswfalse
```

*(End definition for \@iffilesw and \@ifpartsw.)*

\@clubpenalty This stores the current normal (non-infinite) value of \clubpenalty; it should therefore be reset whenever the normal value is changed (as in the bibliography in the standard styles).

```
7 \newcount\@clubpenalty
8 \@clubpenalty \clubpenalty
```

*(End definition for \@clubpenalty.)*

```
\document
9 </2ekernel>
10 <latexrelease>\IncludeInRelease{2020/10/01}%
11 <latexrelease> {\document}{Added hook to load l3backend code}%
12 <2ekernel | latexrelease>
13 \def\document{%
```

We do cancel the grouping as part of the \begin handling (this is now done inside \begin instead) so that the env//<env>/begin hook is not hidden inside \begingroup ... \endgroup.

```
14 % \endgroup
15 \UseOneTimeHook{begindocument/before}-%
16 \@kernel@after@begindocument@before
```

Added hook to load l3backend code:

```
17 \ExplSyntaxOn
18 \ifx\@unusedoptionlist\empty\noelse
19 \@latex@warning@no@line{Unused global option(s):`^J%
20 \spaces[\@unusedoptionlist]}%
21 \fi
22 \colht\textheight
23 \colroom\textheight \vsize\textheight
24 \columnwidth\textwidth
25 \clubpenalty\clubpenalty
26 \if@twocolumn
27 \advance\columnwidth -\columnsep
```

```

28 \divide\columnwidth\tw@ \hsize\columnwidth \iffirstcolumntrue
29 \fi
30 \hsize\columnwidth \linewidth\hsize
31 \begingroup\@floatplacement\@dblfloatplacement
32 \makeatletter\let\@writefile\gobbletwo
33 \global \let \multiplelabels \relax
34 \cinput{\jobname.aux}%
35 \endgroup
36 \if@files
37 \immediate\openout\mainaux\jobname.aux
38 \immediate\write\mainaux{\relax}%
39 \fi

```

Dateline 1991/03/26: FMi added `\process@table` to support NFSS; This will also work with old lfonts if no other style defines `\process@table`. The following line forces the initialization of the math fonts.

```

40 \process@table
41 \let\glb@currsize\empty % Force math initialization.

42 \normalsize
43 \everypar{}%

```

So that punctuation in headings is not disturbed by verbatim or other local changes to the space factor codes, save the document default here. This will be locally reset by the output routine. For special cases a class may want to define `\normalsfcodes` directly, in case that definition will be used. (This is an old bug, problem existed in L<sup>A</sup>T<sub>E</sub>X2.0x and plain T<sub>E</sub>X.)

```

44 \ifx\normalsfcodes\empty
45 \ifnum\sfcodes`_=\@m
46 \let\normalsfcodes\frenchspacing
47 \else
48 \let\normalsfcodes\nonfrenchspacing
49 \fi
50 \fi

```

For similar reasons also save the default language, this will be reset locally in the output routine. In particular it allows hyphenation in the page head even if the page break happens in verbatim. If this has already been set by a package, set to the value of `\language` at this point.

```

51 \ifx\document@default@\language\@ne
52 \chardef\document@default@\language\language
53 \fi

```

Way back in 1991 (08/26) FMi & RmS set the `\@noskipsec` switch to true in the preamble and to false here. This was done to trap lists and related text in the preamble but it does not catch everything; hence Change 1.1g was introduced.

```

54 \@noskipsecfalse
55 \let \@refundefined \relax

```

Just before disabling the preamble commands we execute the begin document hook which contains any code contributed by `\AtBeginDocument`. Also disable the gathering of the file list, if no `\listfiles` has been issued. `\AtBeginDocument` is redefined at this point so that and such commands that get into the hook do not chase their tail...

```

56 \@kernel@before@begindocument
57 \UseOneTimeHook{begindocument}%
58 \@kernel@after@begindocument

```

Most of the following assignments will be done globally in case the user adds something like `\begin{multicols}` to the document hook, i.e. starts are group in `\begin{document}`.

Since a value of exactly 0pt for `\topskip` causes `\twocolumn[]` to misbehave, we add this check, hoping that it will not cause any problems elsewhere.

```

59 \ifdim\topskip<1sp\global\topskip 1sp\relax\fi
60 \global\@maxdepth\maxdepth
61 \global\let\@begindocumenthook\@undefined
62 \ifx\@listfiles\@undefined
63 \global\let\@filelist\relax
64 \global\let\@addtofilelist\@gobble
65 \fi

```

At the very end we disable all preamble commands. This has to happen after the begin document hooks was executed so that this hook can still use such commands.

```

66 \gdef\do##1{\global\let ##1\@notprerr}%
67 \@preamblecmds

```

The next line saves tokens and also allows `\@nодокумент` to be used directly to trap preamble errors.

```
68 \global\let \@nодокумент \relax
```

The next line is a pure safety measure in case a do list is ever expanded at the wrong place. In addition it will save a few tokens to get rid of the above definition.

```

69 \global\let\do\noexpand
70 \UseOneTimeHook{begindocument/end}%

```

Use of the hook might mean that we are already in horizontal mode, so ignore the space after `\begin{document}`.

```
71 \ignorespaces}
```

The `begindocument` hook already existed in the kernel since 1994 under the name `\atbegindocumenthook` the additional ones are originally from the `etoolbox` package under the names `\@endpreamblehook` `\afterpreamble`.

```

72 \NewHook{begindocument}
73 \NewHook{begindocument/before}
74 \NewHook{begindocument/end}

```

Above we used two kernel only hooks to be run after the public `begindocument/before` and after `begindocument` hooks.

In `\@kernel@after@begindocument@before` we already place one action: drop the fast execution code for the `env/document/begin` hook. That hook marks the end of the preamble and should therefore only be run once. In a normal document that is anyway the case (so the code would just sit there taking up space afterwards, which these days is rather harmless), however, in more complicated scenarios where several full documents are combined to a single document it might get applied several times with harmful effects. We therefore explicitly drop it at this point. the coding is somewhat obscure due to the name of the macro which requires constructing.

```

75 \edef \@kernel@after@begindocument@before {%
76 \let\expandafter\noexpand\csname

```

```

77 __hook env/document/begin\endcsname
78 \noexpand\@empty}
79 \%let \@kernel@before@begindocument \@empty
80 \%let \@kernel@after@begindocument \@empty

81 </2ekernel | latexrelease>
82 <latexrelease>\EndIncludeInRelease

83 <| latexrelease>\IncludeInRelease{2017/04/15}%
84 <| latexrelease> {\document}{Save language for hyphenation}%
85 <| latexrelease>
86 <| latexrelease>\def\document{\endgroup
87 <| latexrelease> \ifx\@unusedoptionlist\@empty\else
88 <| latexrelease> \@latex@warning@no@line{Unused global option(s):`^J'%
89 <| latexrelease> \@spaces[\@unusedoptionlist]}%
90 <| latexrelease> \fi
91 <| latexrelease> \@colht\textheight
92 <| latexrelease> \@colroom\textheight \vsize\textheight
93 <| latexrelease> \columnwidth\textwidth
94 <| latexrelease> \clubpenalty\clubpenalty
95 <| latexrelease> \if@twocolumn
96 <| latexrelease> \advance\columnwidth -\columnsep
97 <| latexrelease> \divide\columnwidth\tw@ \hsize\columnwidth \@firstcolumntrue
98 <| latexrelease> \fi
99 <| latexrelease> \hsize\columnwidth \linewidth\hsize
100 <| latexrelease> \begingroup\@floatplacement\@dblfloatplacement
101 <| latexrelease> \makeatletter\let\@writefile\@gobbletwo
102 <| latexrelease> \global\let\@multiplelabels\relax
103 <| latexrelease> \@input{\jobname.aux}%
104 <| latexrelease> \endgroup
105 <| latexrelease> \if@filesaw
106 <| latexrelease> \immediate\openout\@mainaux\jobname.aux
107 <| latexrelease> \immediate\write\@mainaux{\relax}%
108 <| latexrelease> \fi
109 <| latexrelease> \process@table
110 <| latexrelease> \let\glb@currsize\@empty % Force math initialization.
111 <| latexrelease> \normalsize
112 <| latexrelease> \everypar{}%
113 <| latexrelease> \ifx\normalsfcodes\@empty
114 <| latexrelease> \ifnum\sfcode`.=\@m
115 <| latexrelease> \let\normalsfcodes\frenchspacing
116 <| latexrelease> \else
117 <| latexrelease> \let\normalsfcodes\nonfrenchspacing
118 <| latexrelease> \fi
119 <| latexrelease> \fi
120 <| latexrelease> \ifx\document@default@language\m@ne
121 <| latexrelease> \chardef\document@default@language\language
122 <| latexrelease> \fi
123 <| latexrelease> \noskipsecfalse
124 <| latexrelease> \let\@refundefined\relax
125 <| latexrelease> \let\AtBeginDocument\@firstofone
126 <| latexrelease> \begindocumenthook

```

```

127 <|latexrelease> \ifdim\topskip<1sp\global\topskip 1sp\relax\fi
128 <|latexrelease> \global\@maxdepth\maxdepth
129 <|latexrelease> \global\let\@begindocumenthook\@undefined
130 <|latexrelease> \ifx\@listfiles\@undefined
131 <|latexrelease> \global\let\@filelist\relax
132 <|latexrelease> \global\let\@addtofilelist\gobble
133 <|latexrelease> \fi
134 <|latexrelease> \gdef\do##1{\global\let ##1\@notprerr}%
135 <|latexrelease> \@preamblecmds
136 <|latexrelease> \global\let \@nodocument \relax
137 <|latexrelease> \global\let\do\noexpand
138 <|latexrelease> \ignorespaces}
139 <|latexrelease>\EndIncludeInRelease
140 <|latexrelease>
141 <|latexrelease>\IncludeInRelease{0000/00/00}%
142 <|latexrelease> {\document}{Save language for hyphenation}
143 <|latexrelease>\def\document{\endgroup
144 <|latexrelease> \ifx\@unusedoptionlist\@empty\else
145 <|latexrelease> \@latex@warning@no@line{Unused global option(s):`^J}%
146 <|latexrelease> \@spaces[\@unusedoptionlist]}%
147 <|latexrelease> \fi
148 <|latexrelease> \@colht\textheight
149 <|latexrelease> \@colroom\textheight \vsize\textheight
150 <|latexrelease> \columnwidth\textwidth
151 <|latexrelease> \clubpenalty\clubpenalty
152 <|latexrelease> \if@twocolumn
153 <|latexrelease> \advance\columnwidth -\columnsep
154 <|latexrelease> \divide\columnwidth\tw@ \hsize\columnwidth
155 <|latexrelease> \firstcolumntrue
156 <|latexrelease> \fi
157 <|latexrelease> \hsize\columnwidth \linewidth\hsize
158 <|latexrelease> \begingroup\@floatplacement\@dblfloatplacement
159 <|latexrelease> \makeatletter\let\@writefile\@gobbletwo
160 <|latexrelease> \global\let \@multiplelabels \relax
161 <|latexrelease> \@input{\jobname.aux}%
162 <|latexrelease> \endgroup
163 <|latexrelease> \if@files
164 <|latexrelease> \immediate\openout\@mainaux\jobname.aux
165 <|latexrelease> \immediate\write\@mainaux{\relax}%
166 <|latexrelease> \fi
167 <|latexrelease> \process@table
168 <|latexrelease> \let\glb@currsize\@empty
169 <|latexrelease> \normalsize
170 <|latexrelease> \everypar{}%
171 <|latexrelease> \ifx\normalsfcodes\@empty
172 <|latexrelease> \ifnum\sfcodes`.=\@m
173 <|latexrelease> \let\normalsfcodes\frenchspacing
174 <|latexrelease> \else
175 <|latexrelease> \let\normalsfcodes\nonfrenchspacing
176 <|latexrelease> \fi
177 <|latexrelease> \fi
178 <|latexrelease> \noskipsecfalse
179 <|latexrelease> \let \@refundefined \relax
180 <|latexrelease> \let\AtBeginDocument\@firstofone

```

```

181 〈latexrelease〉 \@begindocumenthook
182 〈latexrelease〉 \ifdim\topskip<1sp\global\topskip 1sp\relax\fi
183 〈latexrelease〉 \global\@maxdepth\maxdepth
184 〈latexrelease〉 \global\let\@begindocumenthook\@undefined
185 〈latexrelease〉 \ifx\@listfiles\@undefined
186 〈latexrelease〉 \global\let\@filelist\relax
187 〈latexrelease〉 \global\let\@addtofilelist\@gobble
188 〈latexrelease〉 \fi
189 〈latexrelease〉 \gdef\do##1{\global\let ##1\@notprerr}%
190 〈latexrelease〉 \@preamblecmds
191 〈latexrelease〉 \global\let \nодокумент \relax
192 〈latexrelease〉 \global\let\do\noexpand
193 〈latexrelease〉 \ignorespaces}
194 〈latexrelease〉\EndIncludeInRelease
195 〈*2ekernel〉
196 \onlypreamble\document

```

(*End definition for \document and others.*)

- \normalsfcodes** The setting of \empty is just a flag. This command may be defined in a class or package file. If it is still \empty at \begin{document} it will be defined to be \frenchspacing or \nonfrenchspacing, depending on which of those appears to be in effect at that point.

```
197 \let\normalsfcodes\empty
```

(*End definition for \normalsfcodes.*)

- \nofiles** Set \filesfalse which suppresses the places where LATEX makes \immediate writes. The \makeindex and \makeglossary are disabled. \protected@write is redefined not to write to the file specified, but rather to write a blank line to the log file. This ensures that a *whatsit* node is still created, and so spacing is not affected by the \nofiles command; to ensure this more generally, the \ifnobreak test is needed.

```

198 \def\nofiles{%
199 \filesfalse
200 \typeout{No auxiliary output files.^J}%
201 \long\def\protected@write##1##2##3{%
202 {\write\m@ne{}\ifnobreak\ifvmode\nobreak\fi\fi}%
203 \let\makeindex\relax
204 \let\makeglossary\relax
205 \onlypreamble\nofiles

```

(*End definition for \nofiles.*)

- \protected@write** This takes three arguments: an output stream, some initialization code, and some text to write. It then writes this, with appropriate handling of \protect and \thepage.

```

206 \long\def \protected@write#1#2#3{%
207 \begingroup
208 \let\thepage\relax
209 #2%
210 \let\protect\@unexpandable@protect
211 \edef\reserved@a{\write#1{#3}}%
212 \reserved@a
213 \endgroup
214 \ifnobreak\ifvmode\nobreak\fi\fi
215 }

```

(End definition for \protected@write.)

216 \let\@auxout=\@mainaux  
\include In the definition of \include, \def\reserved@b changed to \edef\reserved@b to be  
\includeonly consistent with the \edef in \includeonly. (Suggested by Rainer Schöpf & Frank  
Mittelbach. Change made 20 Jul 88.)

Changed definition of \include to allow space at end of file name — otherwise,  
typing \include{foo } would cause L<sup>A</sup>T<sub>E</sub>X to overwrite foo.tex. Change made 24 May  
89, suggested by Rainer Schöpf and Frank Mittelbach

Made \include check for being used inside an \include'd file, as this will not work  
and cause surprising results.

```
217 〈/2ekernel〉
218 〈*2ekernel | latexrelease〉
219 〈latexrelease〉\IncludeInRelease{2020/10/01}%">
220 〈latexrelease〉\{\includeonly\}{Spaces in file names}%">
221 \def\include#1{\relax
222 \ifnum\@auxout=\@partaux
223 \@latex@error{\string\include\space cannot be nested}\@eha
224 \else
```

Here the normalization will add .tex for all files, (it uses the same normalization as the  
hooks), so we need to remove that manually. \@strip@tex@ext does that.

```
225 \set@curr@file{#1}%">
226 \edef\@curr@file{\@strip@tex@ext\@curr@file}%">
227 \expandafter\@include\expandafter{\@curr@file} % deliberate space
228 \fi}
```

Here in \includeonly we also need to strip .tex after normalization:

```
229 \def\includeonly#1%
230 \@partswtrue
```

Because the argument to \includeonly is a comma-separated list of filenames where  
there may be comma's preceding some of the filenames or trailing them. Therefore we  
need to take the list apart, remove the unwanted spaces while leaving the spaces *in* the  
filenames intact.

```
231 \let\@partlist\@empty
232 \@for\reserved@a:=#1 \do
233 {
234 \expandafter\set@curr@file\expandafter{\reserved@a}%">
235 \ifx\@partlist\@empty
236 \edef\@partlist{\@strip@tex@ext\@curr@file}%">
237 \else
238 \edef\@partlist{\@partlist,\@strip@tex@ext\@curr@file}%">
239 \fi
240 }%
241 }
242 \@onlypreamble\includeonly
```

(End definition for \include and \includeonly.)

\@strip@tex@ext These macros take a (\detokenized file name and remove any .tex extension). Extra care is taken to not remove the string .tex from the middle of a file name: it is only removed if it's the very last thing in the file name.

```

243 \def\reserved@a#1{%
244 \def\@strip@tex@ext##1{%
245 \expandafter\@strip@tex@ext@aux
246 ##1\@nil\@nil
247 #1\@nil\relax\@nnil}
248 \def\@strip@tex@ext@aux##1#1\@nil##2\@nnil{%
249 \ifx\relax##2\@empty
250 \expandafter\@cdr\expandafter\@empty\@cdr{}##1%
251 \else##1\fi}%
252 \expandafter\reserved@a
253 \expandafter{\detokenize{.tex}}
254
```

(End definition for \@strip@tex@ext and \@strip@tex@ext@aux.)

```

255 <|latexrelease>\EndIncludeInRelease
256 <|latexrelease>\IncludeInRelease{2019/10/01}%
257 <|latexrelease> {\includeonly}{Spaces in file names}%
258 <|latexrelease>
259 <|latexrelease>\def\includeonly#1{%
260 <|latexrelease> \@partswtrue
261 <|latexrelease> \set@curr@file{\zap@space#1 \@empty}%
262 <|latexrelease> \let\@partlist\@curr@file
263 <|latexrelease> }
264 <|latexrelease>
265 <|latexrelease>\def\include#1{\relax
266 <|latexrelease> \ifnum\@auxout=\@partaux
267 <|latexrelease> \@latex@error{\string\include\space cannot be nested}\@eha
268 <|latexrelease> \else
269 <|latexrelease> \set@curr@file{#1 }%
270 <|latexrelease> \expandafter\@include\@curr@file
271 <|latexrelease> \fi}
272 <|latexrelease>
273 <|latexrelease>\let\@strip@tex@ext\@undefined
274 <|latexrelease>\let\@strip@tex@ext@aux\@undefined
275 <|latexrelease>
276 <|latexrelease>\EndIncludeInRelease
277 <|latexrelease>\IncludeInRelease{0000/00/00}%
278 <|latexrelease> {\includeonly}{Spaces in file names}%
279 <|latexrelease>\def\includeonly#1{%
280 <|latexrelease> \@partswtrue
281 <|latexrelease> \edef\@partlist{\zap@space#1 \@empty}%
282 <|latexrelease>
283 <|latexrelease>\def\include#1{\relax
284 <|latexrelease> \ifnum\@auxout=\@partaux
285 <|latexrelease> \@latex@error{\string\include\space cannot be nested}\@eha
286 <|latexrelease> \else \@include#1 \fi}
287 <|latexrelease>
288 <|latexrelease>\EndIncludeInRelease
289
```

(\*2ekernel)

```

290 </2ekernel>
291 {*2ekernel | latexrelease}
292 {latexrelease}\IncludeInRelease{2020/10/01}%
293 {latexrelease} {\@include}{Spaces in file names and hooks}%

294 \def\@include#1 {%
295 \ifx\@nodocument\relax
296 \clearpage
297 \if@filesw
298 \immediate\write\@mainaux{\string\@input{#1.aux}}%
299 \fi
300 \tempswattrue
301 \if@partsw
302 \tempswafalse
303 \edef\reserved@b{#1}%
304 \for\reserved@a:=\partlist\do
305 {\ifx\reserved@a\reserved@b\tempswattrue\fi}%
306 \fi
307 \if@tempswa
308 \let\auxout\partaux
309 \if@filesw
310 \immediate\openout\partaux "#1.aux"
311 \immediate\write\partaux{\relax}%
312 \fi

```

Now before going to the hooks we need to set `\CurrentFile`:

```

313 %-----%
314 \filehook@set@CurrentFile

```

Execute the `before` hooks just after we switched the `.aux` file ...

```

315 \UseHook{include/before}%
316 \UseOneTimeHook{include/#1/before}%
317 %-----%
318 \input{#1.tex}%
319 %-----%
... then end hooks ...
320 \UseOneTimeHook{include/#1/end}%
321 \UseHook{include/end}%
322 %-----%
323 \clearpage
324 %-----%

```

... and after the `\clearpage` the `after` hooks followed by another `\clearpage` just in case new material got added (after all we need to be in well defined state after the `\include`).

```

325 \UseOneTimeHook{include/#1/after}%
326 \UseHook{include/after}%
327 \clearpage
328 %-----%
329 \writeckpt{#1}%
330 \if@filesw
331 \immediate\closeout\partaux

```

```

332 \fi
333 \else
```

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

```

334 \deadcycles\z@%
335 \nameuse{cp@#1}%
336 \fi
337 \let\auxout\mainaux

338 \else
339 \@latex@warning{%
340 \noexpand\include should only be used after \string\begin{document}}%
341 \input{#1}%
342 \fi}
```

Now declare the non-generic `include` hooks used above:

```

343 \NewHook{include/before}
344 \NewReversedHook{include/end}
345 \NewReversedHook{include/after}
346 <latexrelease>\EndIncludeInRelease
347 </2ekernel | latexrelease>

348 <latexrelease>\IncludeInRelease{0000/00/00}%
349 <latexrelease> {\@include}{Spaces in file names}%
350 <latexrelease>\def\@include#1 {%
351 <latexrelease> \clearpage
352 <latexrelease> \if@filesw
353 <latexrelease> \immediate\write\mainaux{\string\@input{#1.aux}}%
354 <latexrelease> \fi
355 <latexrelease> \tempswattrue
356 <latexrelease> \if@partsw
357 <latexrelease> \tempswafalse
358 <latexrelease> \edef\reserved@b{#1}%
359 <latexrelease> \for\reserved@a:=\partlist\do
360 <latexrelease> {\ifx\reserved@a\reserved@b\tempswattrue\fi}%
361 <latexrelease> \fi
362 <latexrelease> \if@tempswa
363 <latexrelease> \let\auxout\partaux
364 <latexrelease> \if@filesw
365 <latexrelease> \immediate\openout\partaux #1.aux
366 <latexrelease> \immediate\write\partaux{\relax}%
367 <latexrelease> \fi
368 <latexrelease> \input{#1.tex}%
369 <latexrelease> \clearpage
370 <latexrelease> \writeckpt{#1}%
371 <latexrelease> \if@filesw
372 <latexrelease> \immediate\closeout\partaux
373 <latexrelease> \fi
374 <latexrelease> \else
375 <latexrelease> \deadcycles\z@%
376 <latexrelease> \nameuse{cp@#1}%
377 <latexrelease> \fi
378 <latexrelease> \let\auxout\mainaux}
379 <latexrelease>
380 <latexrelease>\EndIncludeInRelease
```

```

381 {*2ekernel}

(End definition for \@include.)

\@writeckpt
382 \def\@writeckpt#1{%
383 \if@files
384 \immediate\write\@partaux{\string\@setckpt{#1}\@charlb}%
385 {\let\@elt\@wckptelt \cl@ckpt}%
386 \immediate\write\@partaux{\@charrb}%
387 \fi}
388
(End definition for \@writeckpt.)
```

\@wckptelt

```

388 \def\@wckptelt#1{%
389 \immediate\write\@partaux{%
390 \string\setcounter{#1}{\the\@nameuse{c@#1}}}}
391
(End definition for \@wckptelt.)
```

\@setckpt RmS 93/08/31: introduced \@setckpt

```

391 \def\@setckpt#1{\global\@namedef{cp@#1}}
```

(End definition for \@setckpt.)

\@charlb The following defines \@charlb and \@charrb to be { and }, respectively with \catcode 11.

\@charrb

```

392 {\catcode`[=1 \catcode`]=2
393 \catcode`[=11 \catcode`}=11
394 \gdef\@charlb[{}]
395 \gdef\@charrb[]}
396]% }brace matching
```

(End definition for \@charlb and \@charrb.)

## 1.1 Safe Input Macros

\@curr@file \set@curr@file File name handling is done by generating a csname from the provided file name (which means that UTF-8 octets gets turned into strings as this is what happens if they appear in a csname due to the code in `utf8.def`). By setting \escapchar to -1 we ensure that we don't get a backslash in front. As a result we end up with all characters as catcode 12 (plus spaces). We then sometimes add quotes around the construct (removing any existing inner quotes. Sometimes we only remove the quotes if they have been supplied by the user. There is clearly some room for improvement.

A side effect of the new code is that we will see quotes around file name displays where there haven't been any before.

For compatibility with existing code using `{abc}.tex` or `{one.two}.png`, an initial brace group is discarded before expansion and \string is applied. The content of the brace group is discarded. This means that a leading space will be lost unless protected (by { } or " " or \space) but filenames with a space are hopefully rare.

The definition below is from 2019 and only used during kernel bootstrapping, later on in `ltfilehook.dtx` it will get overwritten.

```

397 \def\set@curr@file#1{%
398 \begingroup
399 \escapechar\m@ne
400 \xdef@\curr@file{%
401 \expandafter\expandafter\expandafter\unquote@name
402 \expandafter\expandafter\expandafter{%
403 \expandafter\string
404 \csname@\firstofone#1\@empty\endcsname}}%
405 \endgroup
406 }

(End definition for \curr@file and \set@curr@file.)
```

\quote@name Quoting spaces  
 \quote@@name  
 \unquote@name

|           |    |         |
|-----------|----|---------|
| a b c     | -> | "a b c" |
| "a b c"   | -> | "a b c" |
| a" "b" "c | -> | "a b c" |
|           | -> | " "     |

```

407 </2ekernel>
408 <*2ekernel | latexrelease>
409 <| latexrelease>\IncludeInRelease{2019/10/01}%
410 <| latexrelease> {\quote@name}{Quote file names}%
411 \def\quote@name#1{"\quote@@name#1\@gobble"}%
412 \def\quote@@name#1"#{\quote@name}
```

and removing quotes ...

```

413 \def\unquote@name#1{\quote@@name#1\@gobble}
```

(End definition for \quote@name, \quote@@name, and \unquote@name.)

\IfFileExists

```

414 \DeclareRobustCommand\IfFileExists[1]{%
415 \set@curr@file{#1}%
416 \expandafter\IfFileExists@\expandafter{\curr@file}}
```

(End definition for \IfFileExists.)

```

417 </2ekernel | latexrelease>
418 <| latexrelease>\EndIncludeInRelease
419 <| latexrelease>\IncludeInRelease{0000/00/00}%
420 <| latexrelease> {\quote@name}{Quote file names}%
421 <| latexrelease>
422 <| latexrelease>\let\quote@name\@undefined
423 <| latexrelease>\let\quote@@name\@undefined
424 <| latexrelease>\let\unquote@name\@undefined
425 <| latexrelease>
426 <| latexrelease>\long\def \IfFileExists#1#2#3{%
427 <| latexrelease> \openin\@inputcheck#1 %
428 <| latexrelease> \ifeof\@inputcheck
429 <| latexrelease> \ifx\input@path\@undefined
430 <| latexrelease> \def\reserved@a{#3}%
431 <| latexrelease> \else
432 <| latexrelease> \def\reserved@a{\iffileonpath{#1}{#2}{#3}}%
433 <| latexrelease> \fi
```

```

434 <|latexrelease> \else
435 <|latexrelease> \closein\@inputcheck
436 <|latexrelease> \edef\@filef@und{\#1 }%
437 <|latexrelease> \def\reserved@a{\#2}%
438 <|latexrelease> \fi
439 <|latexrelease> \reserved@a
440 <|latexrelease>
441 <|latexrelease>\EndIncludeInRelease
442 {*}2ekernel}

```

\IfFileExists@ Argument #1 is \curr@file so catcode 12 string with no quotes.

The original definition picked up arguments #2 and #3 in a way that they couldn't contain unbalanced conditionals. A better implementation would have been not to pick up the arguments at all but instead use the usual \firstoftwo and \secondoftwo. However, that changes how # is interpreted and so we can't do that nowadays without invalidating a lot of code. Therefore the somewhat curious construction near the end.

```

443 </2ekernel>
444 {*}2ekernel | latexrelease>
445 <|latexrelease>\IncludeInRelease{2021/06/01}%
446 <|latexrelease> {\IfFileExists@}{manage unbalanced conditionals}
447 \long\def \IfFileExists@#1#2#3{%
448 \openin\@inputcheck"#1" %
449 \ifeof\@inputcheck
450 \ifx\input@path\@undefined
451 \let\reserved@a\@secondoftwo
452 \else
453 \def\reserved@a{\@iffilenonpath{\#1}}%
454 \fi
455 \else
456 \closein\@inputcheck
457 \edef\@filef@und{"#1 }%
458 \let\reserved@a\@firstoftwo
459 \fi

```

This is just there so that any # inside #2 or #3 needs doubling (as that was the case in the past).

```

460 \expandafter\def\expandafter\reserved@a
461 \expandafter{\reserved@a{\#2}{\#3}}%
462 \reserved@a
463 </2ekernel | latexrelease>
464 <|latexrelease>\EndIncludeInRelease
465 <|latexrelease>\IncludeInRelease{2019/10/01}%
466 <|latexrelease> {\IfFileExists@}{manage unbalanced conditionals}
467 <|latexrelease>
468 <|latexrelease>\long\def \IfFileExists@#1#2#3{%
469 \openin\@inputcheck"#1" %
470 \ifeof\@inputcheck
471 \ifx\input@path\@undefined
472 \def\reserved@a{\#3}%
473 \else
474 \def\reserved@a{\@iffilenonpath{\#1}{\#2}{\#3}}%
475 \fi
476 \else
477 \closein\@inputcheck

```

```

478 〈latexrelease〉 \edef\@filef@und{"#1" }%
479 〈latexrelease〉 \def\reserved@a{#2}%
480 〈latexrelease〉 \fi
481 〈latexrelease〉 \reserved@a
482 〈latexrelease〉\EndIncludeInRelease
483 〈latexrelease〉\IncludeInRelease{0000/00/00}%
484 〈latexrelease〉 {\IfFileExists@}{manage unbalanced conditionals}
485 〈latexrelease〉
486 〈latexrelease〉\let\IfFileExists@\@undefined
487 〈latexrelease〉
488 〈latexrelease〉
489 〈latexrelease〉\EndIncludeInRelease
490 {/2ekernel}

(End definition for \IfFileExists@.)

```

\@iffileonpath If the file is not found by \openin, and \input@path is defined, look in all the directories specified in \input@path.

```

491 {/2ekernel}
492 {/2ekernel | latexrelease}
493 〈latexrelease〉\IncludeInRelease{2019/10/01}%
494 〈latexrelease〉 {\@iffileonpath}{Quote file names}
495 \long\def\@iffileonpath#1{%
496 \let\reserved@a\@secondoftwo
497 \expandafter\@tfor\expandafter\reserved@b\expandafter
498 :\expandafter=\input@path\do{%
499 \openin\@inputcheck\expandafter\quote@name\expandafter{\reserved@b#1} %
500 \ifeof\@inputcheck\else
501 \edef\@filef@und{\expandafter\quote@name\expandafter{\reserved@b#1} }%
502 \let\reserved@a\@firstoftwo%
503 \closein\@inputcheck
504 \@break@tfor
505 \fi}%
506 \reserved@a}

(End definition for \@iffileonpath.)

```

```

507 {/2ekernel | latexrelease}
508 〈latexrelease〉\EndIncludeInRelease
509 〈latexrelease〉\IncludeInRelease{0000/00/00}%
510 〈latexrelease〉 {\@quote@name}{Quote file names}
511 〈latexrelease〉
512 〈latexrelease〉\long\def\@iffileonpath#1{%
513 \let\reserved@a\@secondoftwo
514 \expandafter\@tfor\expandafter\reserved@b\expandafter
515 :\expandafter=\input@path\do{%
516 \openin\@inputcheck\reserved@b#1 %
517 \ifeof\@inputcheck\else
518 \edef\@filef@und{\reserved@b#1 }%
519 \let\reserved@a\@firstoftwo%
520 \closein\@inputcheck
521 \@break@tfor
522 \fi}%
523 \reserved@a}
524 〈latexrelease〉

```

```

525 ⟨latexrelease⟩\EndIncludeInRelease
526 ⟨*2ekernel⟩

\InputIfFileExists Now define \InputIfFileExists to input #1 if it seems to exist. Immediately prior to the input, #2 is executed. If the file #1 does not exist, execute ‘#3’.
This here is a temporary definition for the kernel. The real one comes somewhat later in the file ltfilehook.dtx.
527 \DeclareRobustCommand \InputIfFileExists[2]{%
528 \IfFileExists{#1}{%
529 {%
530 \expandafter\@swaptwoargs\expandafter
531 {\@filef@und}{#2\@addtofilelist{#1}\@@input}}}
532 {%
533 ⟨/2ekernel⟩
534 ⟨*2ekernel | latexrelease⟩
535 ⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
536 ⟨latexrelease⟩{\@swaptwoargs}{Don't lose the file name}%
537 \long\def\@swaptwoargs#1#2{#2#1}
538 ⟨/2ekernel | latexrelease⟩
539 ⟨latexrelease⟩\EndIncludeInRelease
540 ⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
541 ⟨latexrelease⟩{\@swaptwoargs}{Don't lose the file name}%
542 ⟨latexrelease⟩\let\@swaptwoargs\@undefined
543 ⟨/2ekernel⟩
544 }%
545 {%
546 ⟨/2ekernel | latexrelease⟩
547 ⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
548 ⟨latexrelease⟩{\@iinput}{Change in file error handling}%
549 \def\@iinput#1{%
550 \InputIfFileExists{#1}{%
551 {\filename@parse\curr@file
552 \edef\reserved@a{\noexpand\@missingfileerror
553 {\filename@area\filename@base}%
554 {\ifx\filename@ext\relax tex\else\filename@ext\fi}}}}
555 \reserved@a
556 }%
557 }%
558 }
```

(End definition for \InputIfFileExists.)

\@swaptwoargs Swap two arguments and return them unbraced (like \@firstoftwo etc).

```

532 {%
533 ⟨/2ekernel⟩
534 ⟨*2ekernel | latexrelease⟩
535 ⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
536 ⟨latexrelease⟩{\@swaptwoargs}{Don't lose the file name}%
537 \long\def\@swaptwoargs#1#2{#2#1}
538 ⟨/2ekernel | latexrelease⟩
539 ⟨latexrelease⟩\EndIncludeInRelease
540 ⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
541 ⟨latexrelease⟩{\@swaptwoargs}{Don't lose the file name}%
542 ⟨latexrelease⟩\let\@swaptwoargs\@undefined
543 ⟨/2ekernel⟩
544 }%
545 {%
546 ⟨/2ekernel | latexrelease⟩
547 ⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
548 ⟨latexrelease⟩{\@iinput}{Change in file error handling}%
549 \def\@iinput#1{%
550 \InputIfFileExists{#1}{%
551 {\filename@parse\curr@file
552 \edef\reserved@a{\noexpand\@missingfileerror
553 {\filename@area\filename@base}%
554 {\ifx\filename@ext\relax tex\else\filename@ext\fi}}}}
555 \reserved@a
556 }%
557 }%
558 }
```

(End definition for \@swaptwoargs.)

\input Input a file: if the argument is given in braces use safe input macros, otherwise use TeX’s primitive \input command (which is called \@@input in L<sup>A</sup>T<sub>E</sub>X).

```

544 \def\input{\@ifnextchar\bgroup\@iinput\@@input}
```

(End definition for \input.)

\@iinput Define \@iinput (i.e., \input) in terms of \InputIfFileExists.

Changes to \@iinput: adapt to the changes to \@missingfileerror.

```

545 {%
546 ⟨/2ekernel⟩
547 ⟨*2ekernel | latexrelease⟩
548 ⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
549 ⟨latexrelease⟩{\@iinput}{Change in file error handling}%
550 \def\@iinput#1{%
551 \InputIfFileExists{#1}{%
552 {\filename@parse\curr@file
553 \edef\reserved@a{\noexpand\@missingfileerror
554 {\filename@area\filename@base}%
555 {\ifx\filename@ext\relax tex\else\filename@ext\fi}}}}
556 \reserved@a
557 }%
558 }%
```

This line now just sets \@missingfile@⟨part⟩:

```

555 \reserved@a
```

Now here we have to use it. The file here is guaranteed to exist, because `\@missingfileerror` ensures so, but we have to use `\InputIfFileExists` because it executes the file hooks.

```

556 \edef\reserved@a{\noexpand\@iinput{%
557 \@missingfile@area\@missingfile@base.\@missingfile@ext}}%
558 \reserved@a}%
559
```

`//2ekernel | latexrelease`

```

560 \langle latexrelease\rangle\EndIncludeInRelease

561 \langle latexrelease\rangle\IncludeInRelease[2019/10/01]%
562 \langle latexrelease\rangle {\@iinput}{Quote file names}%
563 \langle latexrelease\rangle
564 \langle latexrelease\rangle\def\@iinput#1{%
565 \langle latexrelease\rangle \InputIfFileExists{#1}{}%
566 \langle latexrelease\rangle {\filename@parse@\curr@file
567 \langle latexrelease\rangle \edef\reserved@a{\noexpand\@missingfileerror
568 \langle latexrelease\rangle {\filename@area\filename@base}%
569 \langle latexrelease\rangle {\ifx\filename@ext\relax tex\else\filename@ext\fi}%
570 \langle latexrelease\rangle \reserved@a}%
571 \langle latexrelease\rangle\EndIncludeInRelease

572 \langle latexrelease\rangle\IncludeInRelease[0000/00/00]%
573 \langle latexrelease\rangle {\@iinput}{Quote file names}%
574 \langle latexrelease\rangle\def\@iinput#1{%
575 \langle latexrelease\rangle \InputIfFileExists{#1}{}%
576 \langle latexrelease\rangle {\filename@parse{#1}%
577 \langle latexrelease\rangle \edef\reserved@a{\noexpand\@missingfileerror
578 \langle latexrelease\rangle {\filename@area\filename@base}%
579 \langle latexrelease\rangle {\ifx\filename@ext\relax tex\else\filename@ext\fi}%
580 \langle latexrelease\rangle \reserved@a}%
581 \langle latexrelease\rangle\EndIncludeInRelease
582
```

(*End definition for `\@iinput`.*)

`\@input` Define `\@input` in terms of `\IfFileExists`. So this is a ‘safe input’ command, but the files input are not listed by `\listfiles`.

We don’t want `.aux`, `.toc` files etc be listed by `\listfiles`. However, something like `.bb1` probably should be listed and thus should be implemented not by `\@input`.

```

583 \def\@input#1{%
584 \IfFileExists{#1}{\@input\@filef@und}{\typeout{No file #1.}}}

```

(*End definition for `\@input`.*)

`\@input@` Version of `\@input` that does add the file to `\@filelist`.

```

585 \def\@input@#1{\InputIfFileExists{#1}{}{\typeout{No file #1.}}}

```

(*End definition for `\@input@`.*)

`\@missingfileerror` This ‘error’ command avoids TeX’s primitive missing file loop.

Missing file error. Prompt for a new filename, offering a default extension.

Changes to `\@missingfileerror`: rather than trying to input the file by force, now `\@missingfileerror` just returns three `\@missingfile@<part>` and the caller macro is responsible for doing the right thing with it.

```

586
```

`//2ekernel`

```

587
```

```
588 \langle latexrelease \rangle \IncludeInRelease{2020/10/01}%
589 \langle latexrelease \rangle \{@missingfileerror}{Do not load missing file immediately}%
590 \gdef\@missingfileerror{\#1\#2\%%
591 \typeout{^^J! LaTeX Error: File '\#1.\#2' not found.^^J^^J}%
592 Type X to quit or <RETURN> to proceed,^^J\%
593 or enter new name. (Default extension: \#2)^^J\}%
594 \message{Enter file name: }%
595 {\endlinechar\m@ne
596 \global\read\m@ne to\@gtempa}%
597 \ifx\@gtempa\@empty
```

If the user answers with `<return>`, fallback to the `.tex` file (previously it did nothing).

```
598 \let\@missingfile@area\@empty
599 \let\@missingfile@base\@empty
600 \def\@missingfile@ext{tex}%
601 \else
```

Use `\batchmode\read-1 to \tl` to end the TeX run, same as `expl3` does (it was `\batchmode\@@end` before).

```
602 \def\reserved@b{\batchmode\read-1 to \reserved@a}%
603 \def\reserved@a{x}\ifx\reserved@a\@gtempa\reserved@b\fi
604 \def\reserved@a{X}\ifx\reserved@a\@gtempa\reserved@b\fi
605 \filename@parse\@gtempa
606 \edef\filename@ext{%
607 \ifx\filename@ext\relax#2\else\filename@ext\fi}%
608 \edef\reserved@a{%
```

Only check `\IfFileExists` (it was `\InputIfFileExists`).

```
609 \noexpand\IfFileExists
610 {\filename@area\filename@base.\filename@ext} {%
```

If the file exists, define \@missingfile@*<part>*.

```
611 {\def\noexpand\@missingfile@area{\filename@area}%
612 \def\noexpand\@missingfile@base{\filename@base}%
613 \def\noexpand\@missingfile@ext {\filename@ext}}}%
614 {\noexpand\@missingfileerror
615 {\filename@area\filename@base}{\filename@ext}}}%
616 \reserved@a
617 \fi
618 }
619 </2ekernel | latexrelease>
620 \latexrelease\EndIncludeInRelease
621 <\latexrelease\IncludeInRelease{0000/00/00}%
622 \latexrelease{\@missingfileerror}{Do not load missing file immediately}%
623 \latexrelease
624 \latexrelease\gdef\@missingfileerror#1#2{%
625 \latexrelease \typeout{^^J! LaTeX Error: File '#1.#2' not found.^^J^^J}%
626 \latexrelease Type X to quit or <RETURN> to proceed,^^J%
627 \latexrelease or enter new name. (Default extension: #2)^^J}%
628 \latexrelease \message{Enter file name: }%
629 \latexrelease \endlinechar\m@ne
630 \latexrelease \global\read\m@ne to\gtempa}%
631 \latexrelease \ifx\gtempa\empty
632 \latexrelease \else
633 \latexrelease \def\reserved@a{x}\ifx\reserved@a\gtempa\batchmode\@end\fi
```

```

634 〈\latexrelease〉 \def\reserved@a{X}\ifx\reserved@a\@gtempa\batchmode\@end\fi
635 〈\latexrelease〉 \filename@parse\@gtempa
636 〈\latexrelease〉 \edef\filename@ext{%
637 \ifx\filename@ext\relax\#2\else\filename@ext\fi}%
638 〈\latexrelease〉 \edef\reserved@a{%
639 \noexpand\InputIfFileExists
640 {\filename@area\filename@base.\filename@ext}%
641 {}%
642 {\noexpand\@missingfileerror
643 {\filename@area\filename@base}{\filename@ext}}}%
644 〈\latexrelease〉 \reserved@a
645 〈\latexrelease〉 \fi}
646 〈\latexrelease〉
647 〈\latexrelease〉\EndIncludeInRelease
648 〈*2ekernel〉

```

(End definition for \@missingfileerror.)

\@obsoletefile For compatibility with L<sup>A</sup>T<sub>E</sub>X 2.09 document styles, we distribute files called `article.sty`, `book.sty`, `report.sty`, `slides.sty` and `letter.sty`. These use the command \@obsoletefile, which produces a warning message.

```

649 \def\@obsoletefile#1#2{%
650 \@latex@warning@no@line{inputting '#1' instead of obsolete '#2'}}%
651 \onlypreamble\@obsoletefile

```

## 1.2 Listing files

A list of files input so far. The initial value of \@gobble eats the comma before the first file name.

```
652 \let\@filelist\@gobble
```

Add to the list of files input so far. This ‘real’ definition is only used for ‘cfg’ files during initex. An initial definition of \@gobble has already been set.

```
653 \%def\@addtofilelist#1{\xdef\@filelist{\@filelist,#1}}
```

A preamble command to cause \end{document} to list files input from the main file.

```

\listfiles
654 \def\listfiles{%
655 \let\listfiles\relax
656 \def\@listfiles##1##2##3##4##5##6##7##8##9\@c{%
657 \def\reserved@d{\%}
658 \@tfor\reserved@c:=##1##2##3##4##5##6##7##8\do{%
659 \ifx\reserved@c\reserved@d
660 \edef\filename@area{\filename@area}%
661 \fi}%
662 \def\@dofilelist{%
663 \typeout{^^J *File List*}%
664 \@for\currname:=\@filelist\do{%
665 \filename@parse\currname
666 \edef\reserved@a{%
667 \filename@base.%}
668 \ifx\filename@ext\relax\relax\else\filename@ext\fi}%
669 \expandafter\let\expandafter\reserved@b

```

```

670 \csname ver@\reserved@a\endcsname
671 \expandafter\expandafter\expandafter@listfiles\expandafter
672 \filename@area\filename@base\\\\\\\\\\\\\\\\\\\\\\@@
673 \typeout{%
674 \filename@area\reserved@a
675 \ifx\reserved@b\relax\else\@spaces\reserved@b\fi}%
676 \typeout{ *****^~J}}}

```

The `\@filelist` will be de-activated if `\listfiles` does not appear in the preamble. `\begin{document}` contains code equivalent to the following:

```

\AtBeginDocument{%
 \ifx\@listfiles\undefined
 \let\@filelist\relax
 \let\@addtofilelist\gobble
 \fi}

677 \@onlypreamble\listfiles

\@dofilelist 678 \let\@dofilelist\relax
679 ⟨/2ekernel⟩

(End definition for \@obsoletefile and others.)

```

# File s

## ltoutenc.dtx

### 1 Font encodings

This section of the kernel contains commands for declaring encoding-specific commands, such as accents. It also contains the code for some of the encoding files, including `omlenc.def`, `omsenc.def`, `t1enc.def` and `ot1enc.def` files, which define the OML, OMS, T1 and OT1 encodings, and the `fontenc` package for selecting encodings.

The `fontenc` package has options for encodings, of which the last option is the default encoding. For example, to use the OT2, OT3 and T1 encodings, with T1 as the default, you say:

```
\usepackage[OT2,OT3,T1]{fontenc}
```

The standard kernel set-up loads font encoding files and selects an encoding as follows.

```
\input {omlenc.def}
\input {t1enc.def}
\input {ot1enc.def}
\input {omsenc.def}
\fontencoding{OT1}
```

Note that the files in the standard `inputenc` package depend on this behaviour of the kernel.

The syntax for declaring encoding-specific commands is:

```
\DeclareTextCommand{\command}{{\encoding}}
[{\number} [{\default}]] {{\commands}}
```

This command is like `\newcommand`, except that it defines a command which is specific to one encoding. The resulting command is always robust, even if its definition is fragile. For example, the definition of `\l` in the OT1 encoding is:

```
\DeclareTextCommand{\l}{OT1}{{\@xxxii 1}}
```

`\DeclareTextCommand` takes the same optional arguments as `\newcommand`.

```
\ProvideTextCommand{\command}{{\encoding}}
[{\number} [{\default}]] {{\commands}}
```

This acts like `\DeclareTextCommand`, but does nothing if the command is already defined.

```
\DeclareTextSymbol{\command}{{\encoding}}{\slot}
```

This command defines a text symbol, with a particular slot in that encoding. The commands:

```
\DeclareTextSymbol{\ss}{OT1}{25}
\DeclareTextCommand{\ss}{OT1}{\char25 }
```

have the same effect, but the `\DeclareTextSymbol` is faster.

```
\DeclareTextAccent{\command}{{\encoding}}{\slot}
```

This command declares a text accent. The commands:

```
\DeclareTextAccent{"}{OT1}{127}
\DeclareTextCommand{"}{OT1}{\add@accent {127}}
```

have the same effect.

```
\DeclareTextComposite{<command>}
{<encoding>}{<argument>}{<slot>}
```

This command declares a composite letter, for example in the T1 encoding '\{a} is slot 225, which is declared by:

```
\DeclareTextComposite{'}{T1}{a}{225}
```

The *command* will normally have been declared with `\DeclareTextAccent`, or as a one-argument `\DeclareTextCommand`.

`\DeclareTextComposite` is the most common example of using the more general declaration `\DeclareTextCompositeCommand`, which can define a composite to be an arbitrary piece of text.

```
\DeclareTextCompositeCommand{<command>}
{<encoding>}{<argument>}{<text>}
```

For example, in the OT1 encoding Å has a hand-crafted definition this is declared as follows

```
\DeclareTextCompositeCommand{r}{OT1}{A}
{\leavevmode\setbox\z@\hbox{!}\dimen@\ht\z@\advance\dimen@-1ex%
\rlap{\raise.67\dimen\hbox{\char23}}A}
```

The *command* will normally have been declared with `\DeclareTextAccent`, or as a one-argument `\DeclareTextCommand`.

The commands defined using the above declarations can be used in two ways. Normally they are used by just calling the command in the appropriate encoding, for example `\ss`. However, sometimes you may wish to use a command in an encoding where it is not defined. If the command has no arguments, then you can use it in another encoding by calling `\UseTextSymbol`:

```
\UseTextSymbol{<encoding>}{<command>}
```

For example, `\UseTextSymbol{OT1}{\ss}` has the same effect as:

```
{\fontencoding{OT1}\selectfont\ss}
```

If the command has one argument then you can use it in another encoding by calling `\UseTextAccent`:

```
\UseTextAccent{<encoding>}{<command>}{<text>}
```

For example, if the current encoding is OT2 then `\UseTextAccent{OT1}{'}{a}` has the same effect as:

```
{\fontencoding{OT1}\selectfont'\{\fontencoding{OT2}\selectfont a\}}
```

You can also declare a default definition for a text command, which will be used if the current encoding has no appropriate definition. Such use will also set the definition for this command in the current encoding to equal this default definition; this makes subsequent uses of the command much faster.

```
\DeclareTextCommandDefault{\command}{\definition}
```

For example, the default definition of the command `\textonequarter` (which produces the fraction  $\frac{1}{4}$ ) could be built using math mode:

```
\DeclareTextCommandDefault{\textonequarter}{\ensuremath {\frac{1}{4}}}
```

There is a matching `\Provide` command which will not override an existing default definition:

```
\ProvideTextCommandDefault{\command}{\definition}
```

The most common use for these commands is to use symbols from other encodings, so there are some optimizations provided:

```
\DeclareTextSymbolDefault{\command}{\encoding}
\DeclareTextAccentDefault{\command}{\encoding}
```

are short for:

```
\DeclareTextCommandDefault{\command}
 {\UseTextSymbol{\encoding}{\command}}
\DeclareTextCommandDefault[1]{\command}
 {\UseTextAccent{\encoding}{\command}#1}
```

For example, to make OT1 the default encoding for `\ss` and `\'` you say:

```
\DeclareTextSymbolDefault{\ss}{OT1}
\DeclareTextAccentDefault{\'}{OT1}
```

Note that you can use these commands on any zero- or one-argument commands declared with `\DeclareText*` or `\ProvideText*`, not just those defined using `\DeclareTextSymbol` or `\DeclareTextAccent`.

## 1.1 Removing encoding-specific commands

In some cases encoding definitions are given to provide some limited support since nothing better is available, for example, the definition for `\textdollar` in OT1 is a hack since \$ and £ actually share the same slot in this encoding. Thus if such a glyph becomes available in a different encoding (e.g., TS1) one would like to get rid of the flaky one and make the default definition point to the new encoding. In such a case defining

```
\DeclareTextSymbol{\textdollar}{TS1}{36}
\DeclareTextSymbolDefault{\textdollar}{TS1}
```

is not enough since if typesetting in OT1 L<sup>A</sup>T<sub>E</sub>X will still find the encoding specific definition for OT1 and therefore ignore the new default. Therefore to ensure that in this case the TS1 version is used we have to remove the OT1 declaration:

```
\UndeclareTextCommand{\textdollar}{OT1}
```

Since the \$ sign is a proper glyph in the T1 encoding there is no point removing its definition and forcing L<sup>A</sup>T<sub>E</sub>X to pick up the TS1 version if typesetting in this encoding. However, assume you want to use the variant dollar sign, i.e., \$ for your dollars. In that case you have to get rid of the T1 declaration as well, e.g., the following would do that for you:

```
\UndeclareTextCommand{\textdollar}{OT1}
\UndeclareTextCommand{\textdollar} {T1}
\DeclareTextCommandDefault{\textdollar}
 {\UseTextSymbol{TS1}\textdollaroldstyle}
```

## 1.2 The order of declarations

If an encoding-specific command is defined for more than one encoding, then it will execute fastest in the encoding in which it was defined last since its top-level definition will be set up to execute in that encoding without any overhead.

For this reason the file `fonttext.ltx` currently first loads the definitions for the T1 encoding and then those for the OT1 encoding so that typesetting in OT1 is optimized since that is (still) the default. However, when T1 is explicitly requested (via `\usepackage[T1]{fontenc}`) the top-level definitions are automatically changed to favour T1 since its declarations are reloaded in the process.

For the same reason default declarations should never come last since they are implemented as a special encoding themselves (with the name ?). Specifying them last would simply mean to make those encoding-specific commands equally inefficient in all encodings. Therefore the `textcomp` package, for example, first sets up all defaults to point to TS1 and then declares the commands in the TS1 encoding.

## 1.3 Docstrip modules

This .dtx file is be used to generate several related files containing font encoding definitions. The mutually exclusive docstrip options are listed here.

|          |                                                                                                                                                                                                 |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| T1       | generates <code>t1enc.def</code> for the Cork encoding.                                                                                                                                         |
| TS1      | generates <code>ts1enc.def</code> for the Text Companion encoding.                                                                                                                              |
| TS1sty   | generates <code>textcomp.sty</code> , package that sets up use of the Text Companion encoding.                                                                                                  |
| OT1      | generates <code>ot1enc.def</code> for Knuth's CM encoding.                                                                                                                                      |
| OMS      | generates <code>omsenc.def</code> for Knuth's math symbol encoding.                                                                                                                             |
| OML      | generates <code>omlenc.def</code> for Knuth's math letters encoding.                                                                                                                            |
| OT4      | generates <code>ot4enc.def</code> for the Polish extension to the OT1 encoding, created by B. Jackowski and M. Ry  ko for use with the Polish version of Computer Modern and Computer Concrete. |
| TU       | generates <code>tuenc.def</code> for Unicode font encoding.                                                                                                                                     |
| package  | generates <code>fontenc.sty</code> for selecting encodings.                                                                                                                                     |
| 2ekernel | for the kernel commands.                                                                                                                                                                        |

## 1.4 Definitions for the kernel

### 1.4.1 Declaration commands

This section contains definitions for commands such as accents which depend on the current encoding. These commands will usually be kept in `.def` files, for example `ot1enc.def` contains the definitions for the OT1 encoding.

```
1 {*2ekernel}
2 \message{font encodings,}
Far too many macros in one block here!
```

```
\DeclareTextCommand
\ProvideTextCommand
\DeclareTextSymbol
 @dec@text@cmd
\chardef@text@cmd
 @changed@cmd
 @changed@x
\TextSymbolUnavailable
 @inmathwarn
```

If you say:

```
\DeclareTextCommand{\foo}{T1}...
```

then `\foo` is defined to be `\T1-cmd \foo \T1\foo`, where `\T1\foo` is *one* control sequence, not two! We then call `\newcommand` to define `\T1\foo`.

```
3 \def\DeclareTextCommand{%
4 @dec@text@cmd\newcommand}
5 \def\ProvideTextCommand{%
6 @dec@text@cmd\providecommand}
7 \def@dec@text@cmd#1#2#3{%
8 \expandafter\def\expandafter#2%
9 \expandafter{%
10 \csname#3-cmd\expandafter\endcsname
11 \expandafter#2%
12 \csname#3\string#2\endcsname
13 }%
14 \let@\if definable\@rc@\if definable
15 \expandafter#1\csname#3\string#2\endcsname}
```

This command was introduced to fix a major bug in `@dec@text@cmd` without changing that command itself. This was thought to be necessary because it is defined in more than one package. (Perhaps the more serious bug is to put complex low-level commands like this in packages?)

The problem it solves is that whereas both `\newcommand` and `\providecommand` (used just above) both handle the resetting of `\if definable` (following its disabling in `@dec@text@cmd`), the primitive `\chardef` neither needs the disabling, nor does the resetting.

```
16 \def\chardef@text@cmd{%
17 \let@\if definable\@rc@\if definable
18 \chardef
19 }
20 \def\DeclareTextSymbol#1#2#3{%
21 @dec@text@cmd\chardef@text@cmd#1{#2}#3\relax
22 }
```

The declarations are only available before `\begin{document}`.

```
23 \onlypreamble\DeclareTextCommand
24 \onlypreamble\DeclareTextSymbol
```

The sneaky bit in all this is what `\T1-cmd \foo \T1\foo` does. There are five possibilities, depending on the current values of `\protect`, `\cf@encoding` and `\ifmmode`:

- If `\protect` is `\@typeset@protect` and `\cf@encoding` is T1, then we execute `\T1\foo`. This should be the normal behaviour, and is optimized for speed.
- If `\protect` is `\@typeset@protect`, `\cf@encoding` is (say) OT1, and `\OT1\foo` is defined, then we execute `\OT1\foo`.
- If `\protect` is `\@typeset@protect`, `\cf@encoding` is (say) OT1, we're in text mode, and `\OT1\foo` is undefined, then we define `\OT1\foo` to be the default value of `\foo`, and execute `\OT1\foo`.
- If `\protect` is `\@typeset@protect`, `\cf@encoding` is (say) OT1, we're in math mode, and `\OT1\foo` is undefined, then we execute the default value of `\foo`. (This is necessary so that things like `$X_\copyright$` work properly.)
- If `\protect` is not `\@typeset@protect` then we execute `\noexpand\foo`. For example, if we are writing to a file, then this results in `\foo` being written. If we are in a `\mark`, then `\foo` will be put in the mark—since `\foo` is robust, it will then survive all the things which may happen to it whilst it's a `\mark`.

So after all that, we will either execute the appropriate definition of `\foo` for the current encoding, or we will execute `\noexpand\foo`.

The default value of `\foo` is `\?\foo` if it is defined, and an error message otherwise.

When the encoding is changed from T1 to OT1, `\T1-cmd` is defined to be `\@changed@cmd` and `\OT1-cmd` is defined to be `\@current@cmd`. This means that the test for what the current encoding is can be performed quickly.

```

25 \def\@current@cmd#1{%
26 \ifx\protect\@typeset@protect
27 \cinmathwarn#1%
28 \else
29 \noexpand#1\expandafter\@gobble
30 \fi}

31 \def\@changed@cmd#1#2{%
32 \ifx\protect\@typeset@protect
33 \cinmathwarn#1%
34 \expandafter\ifx\csname\cf@encoding\string#1\endcsname\relax
35 \expandafter\ifx\csname ?\string#1\endcsname\relax
36 \expandafter\def\csname ?\string#1\endcsname{%
37 \TextSymbolUnavailable#1%
38 }%
39 \fi
40 \global\expandafter\let
41 \csname\cf@encoding\string#1\expandafter\endcsname
42 \csname ?\string#1\endcsname
43 \fi
44 \csname\cf@encoding\string#1%
45 \expandafter\endcsname
46 \else
47 \noexpand#1%
48 \fi}

49 \gdef\TextSymbolUnavailable#1{%
50 \@latex@error{%
51 Command \protect#1 unavailable in encoding \cf@encoding%
52 }\@eha}

```

The command `\@inmathwarn` produces a warning message if we are currently in math mode. Note that since this command is used inside text commands, it can't call `\relax` before the `\ifmmode`. This means that it is possible for the warning to fail to be issued at the beginning of a row of an `\halign` whose template enters math mode. This is probably a bad feature, but there's not much that can be done about it, since adding a `\relax` would break ligatures and kerning between text symbols.

A more efficient solution would be to make `\@inmathwarn` and `\@inmatherr` equal to `\empty` and `\relax` by default, and to have `\everymath` reset them to their usual definitions. This is left for future investigation (for example it may break some third party code).

```
53 \def\@inmathwarn#1{%
54 \ifmmode
55 \@latex@warning{Command \protect#1 invalid in math mode}%
56 \fi}
```

(End definition for `\DeclareTextCommand` and others.)

`\DeclareTextCommandDefault`  
`\ProvideTextCommandDefault`

These define commands with encoding ?.

Note that `\DeclareTextCommandDefault` can only be used in the preamble, but that the `\Provide` version is allowed in inputenc .def files, so is allowed anywhere.

```
57 \def\DeclareTextCommandDefault#1{%
58 \DeclareTextCommand#1?}

59 \def\ProvideTextCommandDefault#1{%
60 \ProvideTextCommand#1?}

61 \@onlypreamble\DeclareTextCommandDefault
62 \%@\onlypreamble\ProvideTextCommandDefault
```

They require `\?-cmd` to be initialized as `\@changed@cmd`.

```
63 \expandafter\let\csname?-cmd\endcsname\@changed@cmd
```

(End definition for `\DeclareTextCommandDefault` and `\ProvideTextCommandDefault`.)

`\DeclareTextAccent`

This is just a disguise for defining a TeX `\accent` command.

```
64 \def\DeclareTextAccent#1#2#3{%
65 \DeclareTextCommand#1{#2}{\add@accent{#3}}}

66 \@onlypreamble\DeclareTextAccent
```

(End definition for `\DeclareTextAccent`.)

`\add@accent`

To save space this code is shared between all text accents that are set using the `\accent` primitive. The argument is pre-set in a box so that any font loading that is needed is already done within the box. This is needed because font-loading involves grouping and that would prevent the accent mechanism from working so that the accent would not be positioned over the argument. Declarations that change the font should be allowed (only low-level ones are at present) inside the argument of an accent command, but not size changes, as they involve `\setbox` operations which also inhibit the mechanism of the `\accent` primitive.

Note that the whole process is within a group. For a detailed discussion of this reimplementation and its deficiencies, see pr/3160.

```
67 \def\add@accent#1#2{\hmode@bgroup
```

Turn off the group in `\UseTextSymbol` in case this is used inside the argument of `\add@accent`.

```
68 \let\hmode@start@before@group\@firstofone
69 \setbox\@tempboxa\hbox{\#2%
```

When presetting the argument in a box we record its `\spacefactor` for later use after the accent got typeset. This way something like `\`A` gets the spacefactor of `A` (i.e., 999) rather than the default value of 1000.

```
70 \global\mathchardef\accent@spacefactor\spacefactor}%
```

The accent primitive doesn't allow things `\begingroup` to interfere between accent and base character. Therefore we need to avoid that (they are some hidden inside `\maybe@load@fontshape`). As we don't have to load the fontshape in this case (as that happened in the box above if necessary, we simply disable that part of the code temporarily. We also ignore `\ignorespaces` which has the same issue and may show up as part of `\normalfont` if that is used.

```
71 \let\maybe@load@fontshape\relax
72 \let\ignorespaces\relax
73 \accent#1 #2\egroup\ifmmode\else\spacefactor\accent@spacefactor\fi}
```

Default definition for `\accent@spacefactor` prevents a horrible death of the above macro inside an unprotected `\edef`.

```
74 \let\accent@spacefactor\relax
```

(End definition for `\add@accent`.)

```
\hmode@bgroup
75 \def\hmode@bgroup{\leavevmode\bgroup}
```

(End definition for `\hmode@bgroup`.)

```
\DeclareTextCompositeCommand
 \DeclareTextComposite
 \text@composite
 \text@composite@x
 \strip@args
```

Another amusing game to play with `\expandafter`, `\csname`, and `\string`. When you say `\DeclareTextCompositeCommand{\foo}{T1}{a}{bar}`, we look to see if the expansion of `\T1\foo` begins with `\text@composite`, and if it doesn't, we redefine `\T1\foo` to be:

```
#1 -> \text@composite \T1\foo #1\empty \text@composite {...}
```

where `...` is the previous definition of `\T1\foo`. Finally, we define `\T1\foo-a` to expand to `bar`.

```
76 </2ekernel>
77 <latexrelease>\IncludeInRelease{2017/04/15}{\DeclareTextCompositeCommand}
78 <latexrelease> {test for undeclared accent}%
79 <*2ekernel | latexrelease>
80 \def\DeclareTextCompositeCommand#1#2#3#4{%
81 \expandafter\let\expandafter\reserved@a\csname#2\string#1\endcsname
82 \ifx\reserved@a\relax
83 \DeclareTextCommand#1{#2}{%
84 \@latex@error{\string#1 undeclared in encoding #2}\@eha}%
85 \@latex@info{Composite with undeclared \string#1 in encoding #2}%
86 \expandafter\let\expandafter\reserved@a\csname#2\string#1\endcsname
87 \fi
88 \expandafter\expandafter\expandafter\ifx
89 \expandafter\@car\reserved@a\relax\relax\@nil \text@composite \else
90 \edef\reserved@b##1{%
```

```

1 \def\expandafter\noexpand
2 \csname#2\string#1\endcsname####1{%
3 \noexpand\@text@composite
4 \expandafter\noexpand\csname#2\string#1\endcsname
5 ####1\noexpand\empty\noexpand\@text@composite
6 {##1}}}%
7 \expandafter\reserved@c\expandafter{\reserved@a{##1}}%
8 \fi
9 \expandafter\def\csname\expandafter\string\csname
10 #2\endcsname\string#1-\string#3\empty\endcsname{#4}}%
11 }
12
```

123 </2ekernel | latexrelease>

124 <latexrelease>\EndIncludeInRelease

125 <latexrelease>\IncludeInRelease{0000/00/00}{\DeclareTextCompositeCommand}

126 <latexrelease> {test for undeclared accent}%

127 <latexrelease>\def\DeclareTextCompositeCommand#1#2#3#4{%

128 <latexrelease> \expandafter\let\expandafter\reserved@a
129 <latexrelease> \csname#2\string#1\endcsname
130 <latexrelease> \expandafter\expandafter\expandafter\ifx
131 <latexrelease> \expandafter\@car\reserved@a\relax\relax\@nil
132 <latexrelease> \expandafter\@text@composite \else
133 <latexrelease> \edef\reserved@c{%
134 <latexrelease> \def\expandafter\noexpand
135 <latexrelease> \csname#2\string#1\endcsname####1{%
136 <latexrelease> \noexpand\@text@composite
137 <latexrelease> \expandafter\noexpand\csname#2\string#1\endcsname
138 <latexrelease> ####1\noexpand\empty\noexpand\@text@composite
139 <latexrelease> {##1}}}%
140 <latexrelease> \expandafter\reserved@c\expandafter{\reserved@a{##1}}%
141 <latexrelease> \fi
142 <latexrelease>\expandafter\def\csname\expandafter\string\csname
143 <latexrelease> #2\endcsname\string#1-\string#3\empty\endcsname{#4}}%
144 <latexrelease>\EndIncludeInRelease
145 

146 <\*2ekernel>

147

This all works because:

```
\@text@composite \T1\foo A\@empty \@text@composite {...}
```

expands to `\T1\foo-A` if `\T1\foo-A` has been defined, and `\{\dots\}` otherwise.

Note that `\@text@composite` grabs the first token of the argument and puts just that in the csname. This is so that `\'{\textit{e}}` will work—it checks whether `\T1\'-\textit{e}` is defined (which presumably it isn't) and so expands to `\{accent 1 \textit{e}\}`.

This trick won't always work, for example `\'{{{\itshape e}}}` will expand to (with spaces added for clarity):

```
\csname \string \T1\` - \string {\itshape e} \empty \endcsname
```

which will die pretty horribly. Unfortunately there's not much can be done about this if we're going to use \csname lookups as a fast way of accessing composites.

This has an unfortunate ‘misfeature’ though, which is that in the T1 encoding, `\'{aa}` produces á. This is not the expected behaviour, and should perhaps be fixed if the fix doesn’t affect performance too badly.

Finally, it's worth noting that the `\@empty` is used in `\@text@composite` so that accents will work even when the argument is empty. If you say `\'{}{}` then this looks up `\\"{} - \@empty`, which ought to be `\relax`, and so all is well. If we didn't include the `\@empty`, then `\'{}{}` would expand to:

```
\csname \string \T1\` - \string \endcsname
```

so the `\endcsname` would be `\string`'ed and the whole of the rest of the document would be put inside the `\csname`. This would not be good.

```
126 \def\@text@composite#1#2#3\@text@composite{%
127 \expandafter\@text@composite@x
128 \csname\string#1-\string#2\endcsname}
```

Originally the `\@text@composite@x` macro had two arguments and if #1 was not `\relax` it was executed, otherwise #2 was executed. All this happened within the `\ifx` code so that neither #1 nor #2 could have picked up any additional arguments from the input stream. This has now been changed using the typical `\@firstoftwo / \@secondoftwo` coding. This way the final expansion will happen without any `\else` or `\fi` intervening in the case that we need to get a further token from the input stream.

```
129 \def\@text@composite@x#1{%
130 \ifx#1\relax
131 \expandafter\@secondoftwo
132 \else
133 \expandafter\@firstoftwo
134 \fi
135 #1}
```

The command `\DeclareTextComposite` uses `\DeclareTextCompositeCommand` to declare a command which expands out to a single glyph.

```
136 \catcode\z@=11\relax
137 \def\DeclareTextComposite#1#2#3#4{%
138 \def\reserved@a{\DeclareTextCompositeCommand#1{#2}{#3}}%
139 \bgroup
140 \lccode\z@#4%
141 \lowercase{%
142 \egroup
143 \reserved@a ^^@}}
144 \catcode\z@=15\relax
145 \onlypreamble\DeclareTextComposite
```

*(End definition for `\DeclareTextCompositeCommand` and others.)*

```
146 </2ekernel>
147 (*2ekernel | latexrelease)
148 (latexrelease)\IncludeInRelease{2019/10/01}%
149 (latexrelease) {\UseTextAccent}{Make commands robust}%
```

`\UseTextAccent` These fragile commands access glyphs from different encodings. They use grotty low-level calls to the font selection scheme for speed, and in order to make sure that `\UseTextSymbol` doesn't do anything which you're not allowed to do between an `\accent` and its glyph.

For a detailed discussion of this reimplementation and its deficiencies, see pr/3160.

```

150 \DeclareRobustCommand*\UseTextAccent[3]{%
151 \hmode@start@before@group
152 {%
153 \let\hmode@start@before@group\@firstofone
154 \let\@curr@enc\cf@encoding
155 \use@text@encoding{#1}%
156 #2{\use@text@encoding\@curr@enc#3}%
157 }%
158 \DeclareRobustCommand*\UseTextSymbol[2]{%
159 \hmode@start@before@group
160 {%
161 \def\@wrong@font@char{\MessageBreak
162 for \noexpand\symbol`\'{#2}}%
163 \use@text@encoding{#1}%
164 #2%
165 }%
166 }
167 </2ekernel | latexrelease>
168 <latexrelease>\EndIncludeInRelease
169 <latexrelease>\IncludeInRelease{0000/00/00}%
170 <latexrelease> {\UseTextAccent}{Make commands robust}%
171 <latexrelease>
172 <latexrelease>\kernel@make@fragile\UseTextAccent
173 <latexrelease>\kernel@make@fragile\UseTextSymbol
174 <latexrelease>
175 <latexrelease>\EndIncludeInRelease
176 <*2ekernel>

```

Switch to a different text encoding without any grouping for use in \UseTextAccent or \UseTextSymbol (and for \oldstylenums).

```

177 \def\@use@text@encoding#1{%
178 \edef\f@encoding{#1}%
179 \xdef\font@name{%
180 \csname\curr@fontshape/\f@size\endcsname}%
181 \pickup@font
182 \font@name
183 \@@enc@update}%

```

(End definition for \UseTextAccent, \UseTextSymbol, and \@use@text@encoding.)

\hmode@start@before@group The \hmode@start@before@group starts hmode and should be immediately followed by an explicit \{...\}. Its purpose is to ensure that hmode is started before this group is opened. Inside \add@accent and \UseTextAccent it is redefined to remove this group so that it doesn't conflict with the \accent primitive.

For a detailed discussion see pr/3160.

```

184 \let\hmode@start@before@group\leavevmode

```

(End definition for \hmode@start@before@group.)

|                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \DeclareTextSymbolDefault | Some syntactic sugar. Again, these should probably be optimized for speed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| \DeclareTextAccentDefault | <pre> 185 \def\DeclareTextSymbolDefault#1#2{% 186   \DeclareTextCommandDefault#1{\UseTextSymbol{#2}{#1}}% 187 \def\DeclareTextAccentDefault#1#2{% 188   \DeclareTextCommandDefault#1{\UseTextAccent{#2}{#1}}% 189 \onlypreamble\DeclareTextSymbolDefault 190 \onlypreamble\DeclareTextAccentDefault </pre> <p>(End definition for <code>\DeclareTextSymbolDefault</code> and <code>\DeclareTextAccentDefault</code>.)</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| \UndeclareTextCommand     | <p>This command safely removes an encoding specific declaration for a given encoding. It is helpful if one intends to use the default definition always and therefore wants to get rid of a declaration for some specific encoding.</p> <pre> 191 \def\UndeclareTextCommand#1#2{% </pre> <p>If there is no declaration for the current encoding do nothing. (This makes a hash table entry but without eTeX we can't do anything about that).</p> <pre> 192 \expandafter\ifx\csname#2\string#1\endcsname\relax 193 \else </pre> <p>Else: throw away that declaration.</p> <pre> 194 \global\expandafter\let\csname#2\string#1\endcsname 195 \undefined </pre> <p>But this is unfortunately not enough, we have to take a look at the top-level definition of the encoding specific command which for a command <code>\foo</code> would look similar to <code>\T1-cmd \foo \T1\foo</code> (three tokens).</p> <p>Of course, instead of <code>\T1</code> one could see a different encoding name; which one depends the encoding for which <code>\foo</code> was declared last.</p> <p>Now assume we have just removed the declaration for <code>\foo</code> in <code>\T1</code> and the top-level of <code>\foo</code> expands to the above. Then we better change that pretty fast otherwise we do get an “undefined csname error” when we try to typeset <code>\foo</code> within <code>\T1</code> instead of getting the default definition for <code>\foo</code>. And what is the best way to change that top-level definition? Well, the only “encoding” we know for sure will still be around is the default encoding denoted by <code>?</code>.</p> <p>Thus in case the last token of the top-level expansion is now undefined we change the declaration to look like <code>\?-cmd \foo \?\foo</code> which is done by the following (readable?) code:</p> <pre> 196 \expandafter\expandafter\expandafter 197 \ifx\expandafter\@thirdofthree#1\@undefined 198   \expandafter\gdef\expandafter#1\expandafter 199     {\csname ?-cmd\expandafter\endcsname\expandafter 200       #1\csname?\string#1\endcsname}% 201   \fi 202 \fi 203 } </pre> <pre> 204 \onlypreamble\UndeclareTextCommand </pre> <p>(End definition for <code>\UndeclareTextCommand</code>.)</p> |

### 1.4.2 Hyphenation

```
\patterns
\@@patterns
\hyphenation
\@@hyphenation
205 \%\\let\\@@patterns\\patterns
206 \%\\let\\@@hyphenation\\hyphenation
207 \%\\def\\patterns{%
208 % \\bgroup
209 % \\let\\protect\\empty
210 % \\let\\@typeset\\protect\\empty
211 % \\let\\@changed@x\\@changed@x@mouth
212 % \\afterassignment\\egroup
213 % \\@@patterns
214 %}
215 \%\\def\\hyphenation{%
216 % \\bgroup
217 % \\let\\protect\\empty
218 % \\let\\@typeset\\protect\\empty
219 % \\let\\@changed@x\\@changed@x@mouth
220 % \\afterassignment\\egroup
221 % \\@@hyphenation
222 %}
```

(End definition for `\patterns` and others.)

### 1.4.3 Miscellania

- \a The `\a` command is used to access the accent commands even when they have been redefined (for example by the `tabbing` environment). Its internal name is `\@tabacckludge`.  
The `\string` within the `\csname` guards against something like ' being active at the point of use.

```
223 \\def\\@tabacckludge#1{\\expandafter\\@changed@cmd
224 \\csname\\string#1\\endcsname\\relax}
225 \\let\\a=\\@tabacckludge
```

(End definition for `\a`.)

### 1.4.4 Default encodings

We define the default encodings for most commands to be either OT1, OML or OMS. These defaults are in the kernel and therefore fonts with these encodings must be available unless these defaults are redefined elsewhere. Recall that the standard kernel loads the encoding files for these encodings, and also that for the T1 encoding.

The naming conventions in the kernel are not what we would use if we were starting from scratch... Those defined by DEK (like `\ae` and `\ss`) or by the TeX Users Group Technical Working Group on multi-lingual typesetting (like `\th` and `\ng`) have short names. Those which were added to the kernel in 1993 and early 1994 are named after their Adobe glyph names (like `\guillemotleft` and `\quotedblbase`). Unfortunately, this naming scheme won't work for all glyphs, since some names (like `\space`) are already used, and some (like `\endash`) are very likely to be defined by users. So we're now using the naming scheme of `\text` followed by the Adobe name, (like `\textendash` and `\textsterling`). Except that some glyphs don't have Adobe names, so we're using the names used by fontinst for those (like `\textcompwordmark`). Sigh.

Some accents from OT1:

```
226 \DeclareTextAccentDefault{\\"}{OT1}
227 \DeclareTextAccentDefault{\'}{OT1}
228 \DeclareTextAccentDefault{\.{}}{OT1}
229 \DeclareTextAccentDefault{\=}{OT1}
230 \DeclareTextAccentDefault{\H}{OT1}
231 \DeclareTextAccentDefault{\^}{OT1}
232 \DeclareTextAccentDefault{\`}{OT1}
233 \DeclareTextAccentDefault{\b}{OT1}
234 \DeclareTextAccentDefault{\c}{OT1}
235 \DeclareTextAccentDefault{\d}{OT1}
236 \DeclareTextAccentDefault{\r}{OT1}
237 \DeclareTextAccentDefault{\u}{OT1}
238 \DeclareTextAccentDefault{\v}{OT1}
239 \DeclareTextAccentDefault{\~}{OT1}
```

Some symbols from OT1:

```
240 \% \DeclareTextSymbolDefault{\AA}{OT1}
241 \DeclareTextSymbolDefault{\AE}{OT1}
242 \DeclareTextSymbolDefault{\L}{OT1}
243 \DeclareTextSymbolDefault{\OE}{OT1}
244 \DeclareTextSymbolDefault{\O}{OT1}
245 \% \DeclareTextSymbolDefault{\aa}{OT1}
246 \DeclareTextSymbolDefault{\ae}{OT1}
247 \DeclareTextSymbolDefault{\i}{OT1}
248 \DeclareTextSymbolDefault{\j}{OT1}

249 \DeclareTextSymbolDefault{\ij}{OT1}
250 \DeclareTextSymbolDefault{\IJ}{OT1}

251 \DeclareTextSymbolDefault{\l}{OT1}
252 \DeclareTextSymbolDefault{\oe}{OT1}
253 \DeclareTextSymbolDefault{\o}{OT1}
254 \DeclareTextSymbolDefault{\ss}{OT1}
255 \DeclareTextSymbolDefault{\textdollar}{OT1}
256 \DeclareTextSymbolDefault{\textemdash}{OT1}
257 \DeclareTextSymbolDefault{\textendash}{OT1}
258 \DeclareTextSymbolDefault{\textexclamdown}{OT1}
259 \% \DeclareTextSymbolDefault{\texthphenchar}{OT1}
260 \% \DeclareTextSymbolDefault{\texthphen}{OT1}
261 \DeclareTextSymbolDefault{\textquestiondown}{OT1}
262 \DeclareTextSymbolDefault{\textquotedblleft}{OT1}
263 \DeclareTextSymbolDefault{\textquotedblright}{OT1}
264 \DeclareTextSymbolDefault{\textquotleft}{OT1}
265 \DeclareTextSymbolDefault{\textquotright}{OT1}
266 \DeclareTextSymbolDefault{\textsterling}{OT1}
```

Some symbols from OMS:

```
267 \DeclareTextSymbolDefault{\textasteriskcentered}{OMS}
268 \DeclareTextSymbolDefault{\textbackslash}{OMS}
269 \DeclareTextSymbolDefault{\textbar}{OMS}
270 \DeclareTextSymbolDefault{\textbardbl}{OMS}
271 \DeclareTextSymbolDefault{\textbraceleft}{OMS}
272 \DeclareTextSymbolDefault{\textbraceright}{OMS}
273 \DeclareTextSymbolDefault{\textbullet}{OMS}
```

```

274 \DeclareTextSymbolDefault{\textdaggerdbl}{OMS}
275 \DeclareTextSymbolDefault{\textdagger}{OMS}
276 \DeclareTextSymbolDefault{\textparagraph}{OMS}
277 \DeclareTextSymbolDefault{\textperiodcentered}{OMS}
278 \DeclareTextSymbolDefault{\textsection}{OMS}
279 \DeclareTextAccentDefault{\textcircled}{OMS}

 Some symbols from OML:

280 \DeclareTextSymbolDefault{\textless}{OML}
281 \DeclareTextSymbolDefault{\textgreater}{OML}
282 \DeclareTextAccentDefault{\t}{OML}

 Some defaults we can fake.

 The interface for defining \copyright changed, it used to use \expandafter to add
braces at the appropriate points.

283 \DeclareTextCommandDefault{\textcopyright}{\textcircled{c}}
284 % \expandafter\def\expandafter
285 % \copyright\expandafter{\expandafter{\copyright}}
286 \DeclareTextCommandDefault{\textasciicircum}{\^{}}
287 \DeclareTextCommandDefault{\textasciitilde}{\~{}}
288 \DeclareTextCommandDefault{\textunderscore}{%
289 \leavevmode \kern.06em\vbox{\hrule\@width.3em}}}

 There is no good reason anymore to fake \textcompwordmark.

290 \% \DeclareTextCommandDefault{\textcompwordmark}{\leavevmode\kern\z@}
291 \DeclareTextSymbolDefault{\textcompwordmark}{T1}

292 \DeclareTextCommandDefault{\textvisiblespace}{%
293 \mbox{\kern.06em\vrule\@height.3ex}%
294 \vbox{\hrule\@width.3em}%
295 \hbox{\vrule\@height.3ex}%
}

 Using \fontdimen3 in the next definition is some sort of a kludge (since it is the
interword stretch) but it makes the ellipsis come out right in mono-spaced fonts too (since
there it is zero).

296 \DeclareTextCommandDefault{\textellipsis}{%
297 .\kern\fontdimen3\font
298 .\kern\fontdimen3\font
299 .\kern\fontdimen3\font}

300 \% \DeclareTextCommandDefault{\textregistered}{\textcircled{\scshape r}}
301 \DeclareTextCommandDefault{\textregistered}{\textcircled{%
302 \check@mathfonts\fontsize\sf@size\z@\math@fontsfalse\selectfont R}}
303 \DeclareTextCommandDefault{\texttrademark}{TM}
304 \DeclareTextCommandDefault{\SS}{SS}

305 \DeclareTextCommandDefault{\textordfeminine}{a}
306 \DeclareTextCommandDefault{\textordmasculine}{o}

```

### 1.4.5 Math material

Some commands can be used in both text and math mode:

```
307 \DeclareRobustCommand{\$}{\ifmmode\mathdollar\else\textrm{\$}\fi}
```

We use `\protected` not `\DeclareRobustCommand` so that `\bigl\{` etc. works inside `\protected@edef`.

```
308 \protected\def\{{\ifmmode\lbrace\else\textrm{\{}fi\}}
309 \protected\def\}{\ifmmode\rbrace\else\textrm{\}}fi\}
310 \DeclareRobustCommand{\P}{\ifmmode\mathparagraph\else\textrm{\P}\fi}
311 \DeclareRobustCommand{\S}{\ifmmode\mathsection\else\textrm{\S}\fi}
312 \DeclareRobustCommand{\dag}{\ifmmode\dagger\else\textrm{\dag}\fi}
313 \DeclareRobustCommand{\ddag}{\ifmmode\ddagger\else\textrm{\ddagger}\fi}
```

For historical reasons `\copyright` needs {} around the definition in maths.

```
314 \DeclareRobustCommand{_}{%
315 \ifmmode\nfss@text{\textunderscore}\else\textrm{_}\fi}
316 \DeclareRobustCommand{\copyright}{%
317 \ifmmode{\nfss@text{\textcopyright}}\else\textrm{\copyright}\fi}
318 \DeclareRobustCommand{\pounds}{%
319 \ifmmode\mathsterling\else\textrm{\pounds}\fi}
320 \DeclareRobustCommand{\dots}{%
321 \ifmmode\mathellipsis\else\textrm{\dots}\fi}
322 \let\ldots\dots
```

Default definition of the commabelow accent.

```
323 </2ekernel>
324 <texrel>\IncludeInRelease{2015/10/01}{\textcommabelow}{comma accent}%
325 <2ekernel | texrel>
326 \DeclareTextCommandDefault{\textcommabelow}[1]
327 {\hmode@bgroup\oalign{\null#1\crcr\hidewidth\raise-.31ex
328 \hbox{\check@mathfonts\fontsize\ssf@size\z@
329 \math@fontsfalse\selectfont,\hidewidth}\egroup}
330 <texrel>\EndIncludeInRelease
331 <2ekernel | texrel>
332 <texrel>\IncludeInRelease{0000/00/00}{\textcommabelow}{comma accent}%
333 <texrel>\let\textcommabelow\@undefined
334 <texrel>\expandafter
335 <texrel> \let\csname\string\T1\string\c-G\endcsname\@undefined
336 <texrel>\expandafter
337 <texrel> \let\csname\string\T1\string\c-K\endcsname\@undefined
338 <texrel>\expandafter
339 <texrel> \let\csname\string\T1\string\c-k\endcsname\@undefined
340 <texrel>\expandafter
341 <texrel> \let\csname\string\T1\string\c-L\endcsname\@undefined
342 <texrel>\expandafter
343 <texrel> \let\csname\string\T1\string\c-l\endcsname\@undefined
344 <texrel>\expandafter
345 <texrel> \let\csname\string\T1\string\c-N\endcsname\@undefined
346 <texrel>\expandafter
347 <texrel> \let\csname\string\T1\string\c-n\endcsname\@undefined
348 <texrel>\expandafter
349 <texrel> \let\csname\string\T1\string\c-R\endcsname\@undefined
```

```

350 \let\csname\string\T1\string\c-r\endcsname\@undefined
351 \let\csname\string\T1\string\c-r\endcsname\@undefined
352 \EndIncludeInRelease
 Default definition of the commaabove accent(E.G.).
353 \IncludeInRelease{2016/02/01}{\textcommabov}{comma above}%
354 {*2ekernel | latexrelease}
355 \DeclareTextCommandDefault{\textcommabov}[1]{%
356 \hmode@bgroup
357 \oalign{%
358 \hidewidth
359 \raise.7ex\hbox{%
360 \check@mathfonts\fontsize\ssf@size\z@\math@fontsfalse\selectfont'%
361 }%
362 \hidewidth\crcr
363 \null#1\crcr
364 }%
365 \egroup
366 }
367 \EndIncludeInRelease
368 (/2ekernel | latexrelease)
369 \IncludeInRelease{0000/00/00}{\textcommabov}{comma above}%
370 \let\textcommabov\@undefined
371 \expandafter
372 \let\csname\string\OT1\string\c-g\endcsname\@undefined
373 \expandafter
374 \let\csname\string\T1\string\c-g\endcsname\@undefined
375 \EndIncludeInRelease

```

## 1.5 Definitions for the OT1 encoding

The definitions for the ‘TEX text’ (OT1) encoding.

Declare the encoding.

```

376 {*OT1}
377 \DeclareFontEncoding{OT1}{}{}

```

Declare the accents.

```

378 \DeclareTextAccent{"}{OT1}{127}
379 \DeclareTextAccent{'}{OT1}{19}
380 \DeclareTextAccent{.}{OT1}{95}
381 \DeclareTextAccent{=}{OT1}{22}
382 \DeclareTextAccent{^}{OT1}{94}
383 \DeclareTextAccent{`}{OT1}{18}
384 \DeclareTextAccent{~-}{OT1}{126}
385 \DeclareTextAccent{H}{OT1}{125}
386 \DeclareTextAccent{u}{OT1}{21}
387 \DeclareTextAccent{v}{OT1}{20}
388 \DeclareTextAccent{r}{OT1}{23}

```

Some accents have to be built by hand: Note that `\oalign` and `\o@align` must be inside a group. In these definitions we no longer use the helper function `\sh@ft` from plain.tex since that now has two incompatible definitions.

```

389 \DeclareTextCommand{\b}{OT1}[1]
390 {\hmode@bgroup\o@align{\relax#1\crcr\hidewidth\ltx@sh@ft{-3ex}}%

```

```

391 \vbox to .2ex{\hbox{\char22}\vss}\hidewidth}\egroup}
392 \DeclareTextCommand{\c}{OT1}[1]
393 {\leavevmode\setbox\z@\hbox{\#1}\ifdim\ht\z@=1ex\accent24 #1%
394 \else{\ooalign{\unhbox\z@\crcr\hidewidth\char24\hidewidth}}\fi}
395 \DeclareTextCommand{\d}{OT1}[1]
396 {\hmode@bgroup
397 \o@lign{\relax#1\crcr\hidewidth\ltx@sh@ft{-1ex}. \hidewidth}\egroup}

```

Declare the text symbols.

```

398 \DeclareTextSymbol{\AE}{OT1}{29}
399 \DeclareTextSymbol{\OE}{OT1}{30}
400 \DeclareTextSymbol{\O}{OT1}{31}
401 \DeclareTextSymbol{\ae}{OT1}{26}
402 \DeclareTextSymbol{\i}{OT1}{16}
403 \DeclareTextSymbol{\j}{OT1}{17}
404 \DeclareTextSymbol{\oe}{OT1}{27}
405 \DeclareTextSymbol{\o}{OT1}{28}
406 \DeclareTextSymbol{\ss}{OT1}{25}
407 \DeclareTextSymbol{\textemdash}{OT1}{124}
408 \DeclareTextSymbol{\textendash}{OT1}{123}

```

The `\nobreak\hspace{z@}` is there to prevent a break after the hyphen but allow later breaks in the remainder of the word.

```

409 \DeclareTextCommand{\textnonbreakinghyphen}{OT1}{\mbox{-}\nobreak\hspace{z@}}
410 \DeclareTextCommand{\textfiguredash} {OT1}{\textendash}
411 \DeclareTextCommand{\texthorizontalbar} {OT1}{\textemdash}

```

Using the ligatures helps with OT1 fonts that have `\textexclamdown` and `\textquestiondown` in unusual positions.

```

412 %\DeclareTextSymbol{\textexclamdown}{OT1}{60}
413 %\DeclareTextSymbol{\textquestiondown}{OT1}{62}
414 \DeclareTextCommand{\textexclamdown}{OT1}{!`}
415 \DeclareTextCommand{\textquestiondown}{OT1}{?`}
416 %\DeclareTextSymbol{\texthyphenchar}{OT1}{`-}
417 %\DeclareTextSymbol{\texthyphen}{OT1}{`-}
418 \DeclareTextSymbol{\textquotedblleft}{OT1}{92}
419 \DeclareTextSymbol{\textquotedblright}{OT1}{`}
420 \DeclareTextSymbol{\textquotelleft}{OT1}{`}
421 \DeclareTextSymbol{\textquoteright}{OT1}{`}

```

Some symbols which are faked from others:

```

422 % \DeclareTextCommand{\aa}{OT1}
423 % {{\accent23a}}
424 \DeclareTextCommand{\L}{OT1}
425 {\leavevmode\setbox\z@\hbox{L}\hb@xt@wd\z@{\hss@xxxii L}}
426 \DeclareTextCommand{\l}{OT1}
427 {\hmode@bgroup\@xxxii l\egroup}
428 % \DeclareTextCommand{\AA}{OT1}
429 % {\leavevmode\setbox\z@\hbox{h}\dimen@ht\z@\advance\dimen@-1ex%
430 % \rlap{\raise.67\dimen@\hbox{\char23}}A}

```

In the OT1 encoding Å has a hand-crafted definition, so we have here the first recorded explicit use of `\DeclareTextCompositeCommand`.

```

431 \DeclareTextCompositeCommand{\r}{OT1}{A}
432 {\leavevmode\setbox\z@\hbox{!}\dimen@ht\z@\advance\dimen@-1ex%
433 \rlap{\raise.67\dimen@\hbox{\char23}}A}

```

The dutch language uses the letter ‘ij’. It is available in T1 encoded fonts, but not in the OT1 encoded fonts. Therefore we fake it for the OT1 encoding.

```

434 \DeclareTextCommand{\ij}{OT1}{%
435 \nobreak\hskip\z@skip i\kern-0.02em\nobreak\hskip\z@skip j}
436 \DeclareTextCommand{\IJ}{OT1}{%
437 \nobreak\hskip\z@skip I\kern-0.02em\nobreak\hskip\z@skip J}

```

In the OT1 encoding, £ and \$ share a slot.

```

438 \DeclareTextCommand{\textdollar}{OT1}{\hmode@bgroup
439 \ifdim \fontdimen\@ne\font >\z@
440 \slshape
441 \else
442 \upshape
443 \fi
444 \char`\$\egroup}
445 \DeclareTextCommand{\textsterling}{OT1}{\hmode@bgroup
446 \ifdim \fontdimen\@ne\font >\z@
447 \itshape
448 \else
449 \fontshape{ui}\selectfont
450 \fi
451 \char`\$\egroup}

```

Here we are adding some more composite commands to the OT1 encoding. This makes the use of certain accents with i compatible with their use with the T1 encoding; this enables them to become true L<sup>A</sup>T<sub>E</sub>X internal representations. However, it will make these accents work a little less fast since a check will always be made for the existence of a composite.

```

452 \DeclareTextComposite{\.}{OT1}{i}{`i}
453 \DeclareTextComposite{\.}{OT1}{i}{`i}
454 \DeclareTextCompositeCommand{\`}{OT1}{i}{\@tabacckludge`i}
455 \DeclareTextCompositeCommand{\'}{OT1}{i}{\@tabacckludge'i}
456 \DeclareTextCompositeCommand{\^}{OT1}{i}{^\i}
457 \DeclareTextCompositeCommand{\"}{OT1}{i}{\"i}

```

T1 encoding is given more extensive set of overloads for \c. But here we just adjust \c{g}.

```

458 \ifx\textcommaabove\undefined\else
459 \DeclareTextCompositeCommand{\c}{OT1}{g}{\textcommaabove{g}}
460 \fi
461
```

## 1.6 Definitions for the T1 encoding

The definitions for the ‘Extended T<sub>E</sub>X text’ (T1) encoding.

Declare the encoding.

```

462 <*T1>
463 \DeclareFontEncoding{T1}{}{}

```

Declare the accents.

```

464 \DeclareTextAccent{\`}{T1}{0}
465 \DeclareTextAccent{\'}{T1}{1}
466 \DeclareTextAccent{\^}{T1}{2}

```

```

467 \DeclareTextAccent{\~}{T1}{3}
468 \DeclareTextAccent{\^}{T1}{4}
469 \DeclareTextAccent{\H}{T1}{5}
470 \DeclareTextAccent{\r}{T1}{6}
471 \DeclareTextAccent{\v}{T1}{7}
472 \DeclareTextAccent{\u}{T1}{8}
473 \DeclareTextAccent{\=}{T1}{9}
474 \DeclareTextAccent{\.}{T1}{10}

```

Some accents have to be built by hand. Note that `\o@align` and `\o@lign` must be inside a group. In these definitions we no longer use the helper function `\sh@ft` from plain.tex since that now has two incompatible definitions.

```

475 \DeclareTextCommand{\b}{T1}[1]
476 {\hmode@bgroup\o@lign{\relax#1\crcr\hidewidth\ltx@sh@ft{-3ex}%
477 \vbox to .2ex{\hbox{\char9}\vss}\hidewidth}\egroup}
478 \DeclareTextCommand{\c}{T1}[1]
479 {\leavevmode\setbox\z@\hbox{\#1}\ifdim\ht\z@=1ex\accent11 #1%
480 \else\o@align{\unhbox\z@\crcr
481 \hidewidth\char11\hidewidth}\fi}
482 \DeclareTextCommand{\d}{T1}[1]
483 {\hmode@bgroup
484 \o@lign{\relax#1\crcr\hidewidth\ltx@sh@ft{-1ex}.\hidewidth}\egroup}
485 \DeclareTextCommand{\k}{T1}[1]
486 {\hmode@bgroup\o@align{\null\#1\crcr\hidewidth\char12}\egroup}
487 \DeclareTextCommand{\textogonekcentered}{T1}[1]
488 {\hmode@bgroup\o@align{%
489 \null\#1\crcr\hidewidth\char12\hidewidth}\egroup}

```

Some symbols are constructed.

Slot 24 contains a small circle intended for construction of these two glyphs.

```

490 \DeclareTextCommand{\textperthousand}{T1}
491 {\%\char 24 } % space or 'relax as delimiter?
492 \DeclareTextCommand{\textpertenthousand}{T1}
493 {\%\char 24\char 24 } % space or 'relax as delimiter?

```

For Maltese, `\Hwithstroke` and `\hwithstroke` are needed.

```

494 \DeclareTextCommand{\Hwithstroke}{T1}
495 {%
496 \hmode@bgroup
497 \vphantom{H}%
498 \sbox\z@{H}%
499 \o@align{%
500 H\cr
501 \hidewidth
502 \vrule
503 height \dimexpr 0.7\ht\z@+0.1ex\relax
504 depth -0.7\ht\z@
505 width 0.8\wd\z@
506 \hidewidth\cr
507 }%
508 \egroup
509 }
510 \DeclareTextCommand{\hwithstroke}{T1}
511 {%

```

```

512 \hmode@bgroup
513 \vphantom{h}%
514 \sbox{z@{h}}%
515 \ooalign{%
516 h\cr
517 \kern0.075\wd{z@}
518 \vrule
519 height \dimexpr 0.7\ht{z@}+0.1ex\relax
520 depth -0.7\ht{z@}
521 width 0.4\wd{z@}
522 \hidewidth\cr
523 }%
524 \egroup
525 }
```

Declare the text symbols.

```

526 \%{\ DeclareTextSymbol{\AA}{T1}{197}
527 \ DeclareTextSymbol{\AE}{T1}{198}
528 \ DeclareTextSymbol{\DH}{T1}{208}
529 \ DeclareTextSymbol{\DJ}{T1}{208}
530 \ DeclareTextSymbol{\L}{T1}{138}
531 \ DeclareTextSymbol{\NG}{T1}{141}
532 \ DeclareTextSymbol{\OE}{T1}{215}
533 \ DeclareTextSymbol{\O}{T1}{216}
534 \ DeclareTextSymbol{\SS}{T1}{223}
535 \ DeclareTextSymbol{\TH}{T1}{222}
536 \%{\ DeclareTextSymbol{\aa}{T1}{229}
537 \ DeclareTextSymbol{\ae}{T1}{230}
538 \ DeclareTextSymbol{\dh}{T1}{240}
539 \ DeclareTextSymbol{\dj}{T1}{158}

540 \ DeclareTextSymbol{\guillemetleft}{T1}{19}
541 \ DeclareTextSymbol{\guillemetright}{T1}{20}
542 % old Adobe names
543 \ DeclareTextSymbol{\guillemotleft}{T1}{19}
544 \ DeclareTextSymbol{\guillemotright}{T1}{20}

545 \ DeclareTextSymbol{\guilsinglleft}{T1}{14}
546 \ DeclareTextSymbol{\guilsinglright}{T1}{15}
547 \ DeclareTextSymbol{\i}{T1}{25}
548 \ DeclareTextSymbol{\j}{T1}{26}
549 \ DeclareTextSymbol{\ij}{T1}{188}
550 \ DeclareTextSymbol{\IJ}{T1}{156}
551 \ DeclareTextSymbol{\l}{T1}{170}
552 \ DeclareTextSymbol{\ng}{T1}{173}
553 \ DeclareTextSymbol{\oe}{T1}{247}
554 \ DeclareTextSymbol{\o}{T1}{248}
555 \ DeclareTextSymbol{\quotedblbase}{T1}{18}
556 \ DeclareTextSymbol{\quotesinglbase}{T1}{13}
557 \ DeclareTextSymbol{\ss}{T1}{255}
558 \ DeclareTextSymbol{\textasciicircum}{T1}{`^}
559 \ DeclareTextSymbol{\textasciitilde}{T1}{`~}
560 \ DeclareTextSymbol{\textbackslash}{T1}{``\\`}
561 \ DeclareTextSymbol{\textbar}{T1}{`\|`}
562 \ DeclareTextSymbol{\textbraceleft}{T1}{`\{`}
```

```

563 \DeclareTextSymbol{\textbraceright}{T1}{`}
564 \DeclareTextSymbol{\textcompwordmark}{T1}{23}
565 \DeclareTextSymbol{\textdollar}{T1}{`}
566 \DeclareTextSymbol{\textemdash}{T1}{22}
567 \DeclareTextSymbol{\textendash}{T1}{21}

```

The `\nobreak\hskip\z@` is there to prevent a break after the hyphen but allow later breaks in the remainder of the word.

```

568 \DeclareTextCommand{\textnonbreakinghyphen}{T1}{\mbox{-}\nobreak\hskip\z@}
569 \DeclareTextCommand{\textfiguredash} {T1}{\textendash}
570 \DeclareTextCommand{\texthorizontalbar} {T1}{\textemdash}

571 \DeclareTextSymbol{\textexclamdown}{T1}{189}
572 \DeclareTextSymbol{\textgreater}{T1}{`}
573 %\DeclareTextSymbol{\texthyphenchar}{T1}{127}
574 %\DeclareTextSymbol{\texthyphen}{T1}{`}
575 \DeclareTextSymbol{\textless}{T1}{`}
576 \DeclareTextSymbol{\textquestiondown}{T1}{190}
577 \DeclareTextSymbol{\textquotedblleft}{T1}{16}
578 \DeclareTextSymbol{\textquotedblright}{T1}{17}
579 \DeclareTextSymbol{\textquotedbl}{T1}{`}
580 \DeclareTextSymbol{\textquotelleft}{T1}{`}
581 \DeclareTextSymbol{\textquoteright}{T1}{`}
582 \DeclareTextSymbol{\textsection}{T1}{159}
583 \DeclareTextSymbol{\textsterling}{T1}{191}
584 \DeclareTextSymbol{\textunderscore}{T1}{95}
585 \DeclareTextSymbol{\textvisiblespace}{T1}{32}
586 \DeclareTextSymbol{\th}{T1}{254}

```

Declare the composites.

```

587 \DeclareTextComposite{\.}{T1}{i}{`}
588 \DeclareTextComposite{\.}{T1}{i}{`}

```

"80 = 128

```

589 \DeclareTextComposite{\u}{T1}{A}{128}
590 \DeclareTextComposite{\k}{T1}{A}{129}
591 \DeclareTextComposite{\'}{T1}{C}{130}
592 \DeclareTextComposite{\v}{T1}{C}{131}
593 \DeclareTextComposite{\v}{T1}{D}{132}
594 \DeclareTextComposite{\v}{T1}{E}{133}
595 \DeclareTextComposite{\k}{T1}{E}{134}
596 \DeclareTextComposite{\u}{T1}{G}{135}

```

"88 = 136

```

597 \DeclareTextComposite{\'}{T1}{L}{136}
598 \DeclareTextComposite{\v}{T1}{L}{137}
599 \DeclareTextComposite{\'}{T1}{N}{139}
600 \DeclareTextComposite{\v}{T1}{N}{140}
601 \DeclareTextComposite{\H}{T1}{O}{142}
602 \DeclareTextComposite{\'}{T1}{R}{143}

```

"90 = 144

```

603 \DeclareTextComposite{\v}{T1}{R}{144}
604 \DeclareTextComposite{\'}{T1}{S}{145}
605 \DeclareTextComposite{\v}{T1}{S}{146}
606 \DeclareTextComposite{\c}{T1}{S}{147}

```

```

607 \DeclareTextComposite{\v}{T1}{T}{148}
608 \DeclareTextComposite{\c}{T1}{T}{149}
609 \DeclareTextComposite{\H}{T1}{U}{150}
610 \DeclareTextComposite{\r}{T1}{U}{151}

"A0 = 152
611 \DeclareTextComposite{\"}{T1}{Y}{152}
612 \DeclareTextComposite{\'}{T1}{Z}{153}
613 \DeclareTextComposite{\v}{T1}{Z}{154}
614 \DeclareTextComposite{\.}{T1}{Z}{155}
615 \DeclareTextComposite{\.}{T1}{I}{157}

"A0 = 160
616 \DeclareTextComposite{\u}{T1}{a}{160}
617 \DeclareTextComposite{\k}{T1}{a}{161}
618 \DeclareTextComposite{\'}{T1}{c}{162}
619 \DeclareTextComposite{\v}{T1}{c}{163}
620 \DeclareTextComposite{\v}{T1}{d}{164}
621 \DeclareTextComposite{\v}{T1}{e}{165}
622 \DeclareTextComposite{\k}{T1}{e}{166}
623 \DeclareTextComposite{\u}{T1}{g}{167}

"A8 = 168
624 \DeclareTextComposite{\'}{T1}{l}{168}
625 \DeclareTextComposite{\v}{T1}{l}{169}
626 \DeclareTextComposite{\'}{T1}{n}{171}
627 \DeclareTextComposite{\v}{T1}{n}{172}
628 \DeclareTextComposite{\H}{T1}{o}{174}
629 \DeclareTextComposite{\'}{T1}{r}{175}

"B0 = 176
630 \DeclareTextComposite{\v}{T1}{r}{176}
631 \DeclareTextComposite{\'}{T1}{s}{177}
632 \DeclareTextComposite{\v}{T1}{s}{178}
633 \DeclareTextComposite{\c}{T1}{s}{179}
634 \DeclareTextComposite{\v}{T1}{t}{180}
635 \DeclareTextComposite{\c}{T1}{t}{181}
636 \DeclareTextComposite{\H}{T1}{u}{182}
637 \DeclareTextComposite{\r}{T1}{u}{183}

"B8 = 184
638 \DeclareTextComposite{\"}{T1}{y}{184}
639 \DeclareTextComposite{\'}{T1}{z}{185}
640 \DeclareTextComposite{\v}{T1}{z}{186}
641 \DeclareTextComposite{\.}{T1}{z}{187}

"C0 = 192
642 \DeclareTextComposite{\'}{T1}{A}{192}
643 \DeclareTextComposite{\'}{T1}{A}{193}
644 \DeclareTextComposite{\^}{T1}{A}{194}
645 \DeclareTextComposite{\~}{T1}{A}{195}
646 \DeclareTextComposite{\"}{T1}{A}{196}
647 \DeclareTextComposite{\r}{T1}{A}{197}
648 \DeclareTextComposite{\c}{T1}{C}{199}

```

```

"C8 = 200
649 \DeclareTextComposite{\`}{T1}{E}{200}
650 \DeclareTextComposite{\'}{T1}{E}{201}
651 \DeclareTextComposite{\^}{T1}{E}{202}
652 \DeclareTextComposite{\"}{T1}{E}{203}
653 \DeclareTextComposite{\`}{T1}{I}{204}
654 \DeclareTextComposite{\'}{T1}{I}{205}
655 \DeclareTextComposite{\^}{T1}{I}{206}
656 \DeclareTextComposite{\\"}{T1}{I}{207}

"D0 = 208
657 \DeclareTextComposite{\~}{T1}{N}{209}
658 \DeclareTextComposite{\'}{T1}{O}{210}
659 \DeclareTextComposite{\'}{T1}{O}{211}
660 \DeclareTextComposite{\^}{T1}{O}{212}
661 \DeclareTextComposite{\~}{T1}{O}{213}
662 \DeclareTextComposite{\\"}{T1}{O}{214}

"D8 = 216
663 \DeclareTextComposite{\`}{T1}{U}{217}
664 \DeclareTextComposite{\'}{T1}{U}{218}
665 \DeclareTextComposite{\^}{T1}{U}{219}
666 \DeclareTextComposite{\\"}{T1}{U}{220}
667 \DeclareTextComposite{\'}{T1}{Y}{221}

"E0 = 224
668 \DeclareTextComposite{\`}{T1}{a}{224}
669 \DeclareTextComposite{\'}{T1}{a}{225}
670 \DeclareTextComposite{\^}{T1}{a}{226}
671 \DeclareTextComposite{\~}{T1}{a}{227}
672 \DeclareTextComposite{\\"}{T1}{a}{228}
673 \DeclareTextComposite{\r}{T1}{a}{229}
674 \DeclareTextComposite{\c}{T1}{c}{231}

"E8 = 232
675 \DeclareTextComposite{\`}{T1}{e}{232}
676 \DeclareTextComposite{\'}{T1}{e}{233}
677 \DeclareTextComposite{\^}{T1}{e}{234}
678 \DeclareTextComposite{\\"}{T1}{e}{235}
679 \DeclareTextComposite{\'}{T1}{i}{236}
680 \DeclareTextComposite{\`}{T1}{i}{236}
681 \DeclareTextComposite{\'}{T1}{i}{237}
682 \DeclareTextComposite{\'}{T1}{i}{237}
683 \DeclareTextComposite{\^}{T1}{i}{238}
684 \DeclareTextComposite{\^}{T1}{i}{238}
685 \DeclareTextComposite{\\"}{T1}{i}{239}
686 \DeclareTextComposite{\\"}{T1}{i}{239}

"F0 = 240
687 \DeclareTextComposite{\~}{T1}{n}{241}
688 \DeclareTextComposite{\'}{T1}{o}{242}
689 \DeclareTextComposite{\'}{T1}{o}{243}
690 \DeclareTextComposite{\^}{T1}{o}{244}
691 \DeclareTextComposite{\~}{T1}{o}{245}
692 \DeclareTextComposite{\\"}{T1}{o}{246}

```

```

" F8 = 248

693 \DeclareTextComposite{\`}{T1}{u}{249}
694 \DeclareTextComposite{\'}{T1}{u}{250}
695 \DeclareTextComposite{\^}{T1}{u}{251}
696 \DeclareTextComposite{\"}{T1}{u}{252}
697 \DeclareTextComposite{\'}{T1}{y}{253}

698 \DeclareTextCompositeCommand{\k}{T1}{o}{\textogonekcentered{o}}
699 \DeclareTextCompositeCommand{\k}{T1}{O}{\textogonekcentered{O}}

700 \ifx\textcommaabove\@undefined\else
701 \DeclareTextCompositeCommand{\c}{T1}{g}{\textcommaabove{g}}
702 \fi
703 \ifx\textcommabelow\@undefined\else
704 \DeclareTextCompositeCommand{\c}{T1}{G}{\textcommabelow{G}}
705 \DeclareTextCompositeCommand{\c}{T1}{K}{\textcommabelow{K}}
706 \DeclareTextCompositeCommand{\c}{T1}{k}{\textcommabelow{k}}
707 \DeclareTextCompositeCommand{\c}{T1}{L}{\textcommabelow{L}}
708 \DeclareTextCompositeCommand{\c}{T1}{l}{\textcommabelow{l}}
709 \DeclareTextCompositeCommand{\c}{T1}{N}{\textcommabelow{N}}
710 \DeclareTextCompositeCommand{\c}{T1}{n}{\textcommabelow{n}}
711 \DeclareTextCompositeCommand{\c}{T1}{R}{\textcommabelow{R}}
712 \DeclareTextCompositeCommand{\c}{T1}{r}{\textcommabelow{r}}
713 \fi
714

```

## 1.7 Definitions for the OMS encoding

The definitions for the ‘ $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  math symbol’ (OMS) encoding. Even though this is meant to be a math font, it includes some of the standard  $\mathrm{L}\mathrm{A}\mathrm{T}_{\mathrm{E}}\mathrm{X}$  text symbols.

Declare the encoding.

```

715 {*OMS}
716 \DeclareFontEncoding{OMS}{}{}

```

Declare the symbols. Note that slot 13 has in places been named  $\backslash\mathrm{Orb}$ : please root out and destroy this impurity wherever you find it!

```

717 \DeclareTextSymbol{\textasteriskcentered}{OMS}{3} % "03
718 \DeclareTextSymbol{\textbackslash}{OMS}{110} % "6E
719 \DeclareTextSymbol{\textbar}{OMS}{106} % "6A
720 \DeclareTextSymbol{\textbardbl}{OMS}{107} % "6B
721 \DeclareTextSymbol{\textbraceleft}{OMS}{102} % "66
722 \DeclareTextSymbol{\textbraceright}{OMS}{103} % "67
723 \DeclareTextSymbol{\textbullet}{OMS}{15} % "0F
724 \DeclareTextSymbol{\textdaggerdbl}{OMS}{122} % "7A
725 \DeclareTextSymbol{\textdagger}{OMS}{121} % "79
726 \DeclareTextSymbol{\textparagraph}{OMS}{123} % "7B
727 \DeclareTextSymbol{\textperiodcentered}{OMS}{1} % "01
728 \DeclareTextSymbol{\textsection}{OMS}{120} % "78
729 \DeclareTextSymbol{\textbigcircle}{OMS}{13} % "0D
730 \DeclareTextCommand{\textcircled}{OMS}[1]{\hmode@bgroup
731 \ooalign{\%
732 \hfil\raise .07ex\hbox {\upshape#1}\hfil\crcr
733 \char 13 % "0D

```

```

734 }%
735 \egroup}
736
```

## 1.8 Definitions for the OML encoding

The definitions for the ‘ $\text{\TeX}$  math italic’ (OML) encoding. Even though this is meant to be a math font, it includes some of the standard  $\text{\LaTeX}$  text symbols.

Declare the encoding.

```

737 <*OML>
738 \DeclareFontEncoding{OML}{}{}

```

Declare the symbols.

```

739 \DeclareTextSymbol{\textless}{OML}{`<}
740 \DeclareTextSymbol{\textgreater}{OML}{`>}
741 \DeclareTextAccent{\t}{OML}{127} % "7F
742
```

## 1.9 Definitions for the OT4 encoding

These definitions are for the Polish extension to the ‘ $\text{\TeX}$  text’ (OT1) encoding. This encoding was created by B. Jackowski and M. Ryćko for use with the Polish version of Computer Modern and Computer Concrete. In positions 0–127 it is identical to OT1 but it contains some additional characters in the upper half. The  $\text{\LaTeX}$  support was developed by Mariusz Olko.

The PL fonts that use it are available as follows:

Metafont sources <ftp://ftp.gust.org.pl/TeX/language/polish/pl-mf.zip>;

Font files <ftp://ftp.gust.org.pl/TeX/language/polish/pl-tfm.zip>.

Declare the encoding.

```

743 <*OT4>
744 \DeclareFontEncoding{OT4}{}{}
745 \DeclareFontSubstitution{OT4}{cmr}{m}{n}

```

Declare the accents.

```

746 \DeclareTextAccent{\"}{OT4}{127}
747 \DeclareTextAccent{\'}{OT4}{19}
748 \DeclareTextAccent{\.}{OT4}{95}
749 \DeclareTextAccent{\=}{OT4}{22}
750 \DeclareTextAccent{\^}{OT4}{94}
751 \DeclareTextAccent{\`}{OT4}{18}
752 \DeclareTextAccent{\~}{OT4}{126}
753 \DeclareTextAccent{\H}{OT4}{125}
754 \DeclareTextAccent{\u}{OT4}{21}
755 \DeclareTextAccent{\v}{OT4}{20}
756 \DeclareTextAccent{\r}{OT4}{23}

```

The ogonek accent is available only under a e A & E. But we have to provide some definition for \k. Some other accents have to be built by hand as in OT1:

```

757 \DeclareTextCommand{\k}{OT4}[1]{%
758 \TextSymbolUnavailable{\k{#1}}#1}

```

In these definitions we no longer use the helper function `\sh@ft` from plain.tex since that now has two incompatible definitions.

```

759 \DeclareTextCommand{\b}{OT4}[1]
760 {\hmode@bgroup\o@lign{\relax#1\crcr\hidewidth\ltx@sh@ft{-3ex}%
761 \vbox to .2ex{\hbox{\char22}\vss}\hidewidth}\egroup}
762 \DeclareTextCommand{\c}{OT4}[1]
763 {\leavevmode\setbox\z@\hbox{\#1}\ifdim\ht\z@=1ex\accent24 #1%
764 \else\ooalign{\unhbox\z@\crcr\hidewidth\char24\hidewidth}\fi}
765 \DeclareTextCommand{\d}{OT4}[1]
766 {\hmode@bgroup
767 \o@lign{\relax#1\crcr\hidewidth\ltx@sh@ft{-1ex}. \hidewidth}\egroup}

```

Declare the text symbols.

```

768 \DeclareTextSymbol{\AE}{OT4}{29}
769 \DeclareTextSymbol{\OE}{OT4}{30}
770 \DeclareTextSymbol{\O}{OT4}{31}
771 \DeclareTextSymbol{\L}{OT4}{138}
772 \DeclareTextSymbol{\ae}{OT4}{26}

773 \DeclareTextSymbol{\guillemetleft}{OT4}{174}
774 \DeclareTextSymbol{\guillemetright}{OT4}{175}
775 % old Adobe names
776 \DeclareTextSymbol{\guillemotleft}{OT4}{174}
777 \DeclareTextSymbol{\guillemotright}{OT4}{175}

778 \DeclareTextSymbol{\i}{OT4}{16}
779 \DeclareTextSymbol{\j}{OT4}{17}
780 \DeclareTextSymbol{\l}{OT4}{170}
781 \DeclareTextSymbol{\o}{OT4}{28}
782 \DeclareTextSymbol{\oe}{OT4}{27}
783 \DeclareTextSymbol{\quotedblbase}{OT4}{255}
784 \DeclareTextSymbol{\ss}{OT4}{25}
785 \DeclareTextSymbol{\textemdash}{OT4}{124}
786 \DeclareTextSymbol{\textendash}{OT4}{123}
787 \DeclareTextSymbol{\textexclamdown}{OT4}{60}
788 \% \DeclareTextSymbol{\texthyphenchar}{OT4}{`-}
789 \% \DeclareTextSymbol{\texthyphen}{OT4}{`-}
790 \DeclareTextSymbol{\textquestiondown}{OT4}{62}
791 \DeclareTextSymbol{\textquotedblleft}{OT4}{92}
792 \DeclareTextSymbol{\textquotedblright}{OT4}{`}
793 \DeclareTextSymbol{\textquotel}{OT4}{`}
794 \DeclareTextSymbol{\textquoter}{OT4}{`}

```

Definition for Å as in OT1:

```

795 \DeclareTextCompositeCommand{\r}{OT4}{A}
796 {\leavevmode\setbox\z@\hbox{!}\dimen@\ht\z@\advance\dimen@-1ex%
797 \rlap{\raise.67\dimen@\hbox{\char23}}A}

```

In the OT4 encoding, £ and \$ share a slot.

```

798 \DeclareTextCommand{\textdollar}{OT4}{\hmode@bgroup
799 \ifdim \fontdimen\onefont >\z@
800 \slshape
801 \else
802 \upshape
803 \fi
804 \char`\$\egroup}

```

```

805 \DeclareTextCommand{\textsterling}{OT4}{\hmode@bgroup
806 \ifdim \fontdimen@ne\font >\z@
807 \itshape
808 \else
809 \fontshape{ui}\selectfont
810 \fi
811 \char`\$\egroup}

```

Declare the composites.

```

812 \DeclareTextComposite{\k}{OT4}{A}{129}
813 \DeclareTextComposite{\'}{OT4}{C}{130}
814 \DeclareTextComposite{\k}{OT4}{E}{134}
815 \DeclareTextComposite{\'}{OT4}{N}{139}
816 \DeclareTextComposite{\'}{OT4}{S}{145}
817 \DeclareTextComposite{\'}{OT4}{Z}{153}
818 \DeclareTextComposite{\.}{OT4}{Z}{155}
819 \DeclareTextComposite{\k}{OT4}{a}{161}
820 \DeclareTextComposite{\'}{OT4}{c}{162}
821 \DeclareTextComposite{\k}{OT4}{e}{166}
822 \DeclareTextComposite{\'}{OT4}{n}{171}
823 \DeclareTextComposite{\'}{OT4}{s}{177}
824 \DeclareTextComposite{\'}{OT4}{z}{185}
825 \DeclareTextComposite{\.}{OT4}{z}{187}
826 \DeclareTextComposite{\'}{OT4}{o}{211}
827 \DeclareTextComposite{\'}{OT4}{o}{243}
828
```

## 1.10 Definitions for the TS1 encoding

```

829 <*TS1>
830 \DeclareFontEncoding{TS1}{}{}
831 \DeclareFontSubstitution{TS1}{cmr}{m}{n}

```

Some accents have to be built by hand. Note that \ooalign and \o@lign must be inside a group.

```

832 \DeclareTextCommand{\capitalcedilla}{TS1}[1]
833 {\hmode@bgroup
834 \ooalign{\null#1\crcr\hidewidth\char11\hidewidth}\egroup}
835 \DeclareTextCommand{\capitalogonek}{TS1}[1]
836 {\hmode@bgroup
837 \ooalign{\null#1\crcr\hidewidth\char12\hidewidth}\egroup}

```

Accents for capital letters.

These commands can be used by the end user either directly or through definitions of the type

```
\DeclareTextCompositeCommand{\'}{T1}{X}{\capitalacute X}
```

None of the latter definitions are provided by default, since they are probably rarely used.

```

"00 = 0
838 \DeclareTextAccent{\capitalgrave}{TS1}{0}
839 \DeclareTextAccent{\capitalacute}{TS1}{1}
840 \DeclareTextAccent{\capitalcircumflex}{TS1}{2}
841 \DeclareTextAccent{\capitaltilde}{TS1}{3}
842 \DeclareTextAccent{\capitaldieresis}{TS1}{4}
843 \DeclareTextAccent{\capitalhungarumlaut}{TS1}{5}

```

```

844 \DeclareTextAccent{\capitalring}{TS1}{6}
845 \DeclareTextAccent{\capitalcaron}{TS1}{7}
"08 = 8
846 \DeclareTextAccent{\capitalbreve}{TS1}{8}
847 \DeclareTextAccent{\capitalmacron}{TS1}{9}
848 \DeclareTextAccent{\capitaldotaccent}{TS1}{10}

```

Tie accents.

The tie accent was borrowed from the cmmi font. The tc fonts now provide four tie accents, the first two are done in the classical way with asymmetric glyphs hanging out of their boxes; the new ties are centered in their boxes like all other accents. They need a name: please tell us if you know what to call them.

```

" =
849 \DeclareTextAccent{\t}{TS1}{26}
850 \DeclareTextAccent{\capitaltie}{TS1}{27}
851 \DeclareTextAccent{\newtie}{TS1}{28}
852 \DeclareTextAccent{\capitalnewtie}{TS1}{29}

```

Compound word marks.

The text companion fonts contain two compound word marks of different heights, one has `cap_height`, the other `asc_height`.

```

853 \DeclareTextSymbol{\textcapitalcompwordmark}{TS1}{23}
854 \DeclareTextSymbol{\textascendercompwordmark}{TS1}{31}

```

The text companion symbols.

```

855 \DeclareTextSymbol{\textquotestraightbase}{TS1}{13}
"10 = 16
856 \DeclareTextSymbol{\textquotestraightdblbase}{TS1}{18}
857 \DeclareTextSymbol{\texttwelveudash}{TS1}{21}
858 \DeclareTextSymbol{\textthreequartersemdash}{TS1}{22}
"18 = 24
859 \DeclareTextSymbol{\textleftarrow}{TS1}{24}
860 \DeclareTextSymbol{\textrightarrow}{TS1}{25}
"20 = 32
861 \DeclareTextSymbol{\textblank}{TS1}{32}
862 \DeclareTextSymbol{\textdollar}{TS1}{36}
863 \DeclareTextSymbol{\textquotesingle}{TS1}{39}
"28 = 40

```

The symbol `\textasteriskcentered` “\*” is supposed to be always available in TS1 and that is important as it is used in footnote symbols. However, in a few fonts it is missing even though they are otherwise fairly complete. We therefore use a rather elaborate method and check if the slot has a glyph and if not produce a poor man’s version by using a normal “\*” slightly enlarged and somewhat lowered. The main application for this symbol is in footnote symbols and there it should produce a comparable size and show a similar placement.

```

864 \% \DeclareTextSymbol{\textasteriskcentered}{TS1}{42} % that's wanted
865 \DeclareTextCommand \textasteriskcentered{TS1}{% % and that's needed
866 \iffontchar\font 42 \char42 \else
867 \begingroup\fontencoding{T1}%
868 \fontsize

```

```

869 {\the\dimexpr1.3\dimexpr\f@size pt\relax}%
870 {\f@baselineskip}%
871 \selectfont
872 \raisebox{-0.7ex}{[\dimexpr\height-0.7ex][0pt]{*}}%
873 \endgroup
874 \fi
875 }

Note that '054 is a comma and '056 is a full stop: these make numbers using oldstyle digits easier to input.

876 \DeclareTextSymbol{\textdblhyphen}{TS1}{45}
877 \DeclareTextSymbol{\textfractionsolidus}{TS1}{47}

Oldstyle digits.

'30 = 48

878 \DeclareTextSymbol{\textzerooldstyle}{TS1}{48}
879 \DeclareTextSymbol{\textoneoldstyle}{TS1}{49}
880 \DeclareTextSymbol{\texttwooldstyle}{TS1}{50}
881 \DeclareTextSymbol{\textthreeoldstyle}{TS1}{51}
882 \DeclareTextSymbol{\textfouroldstyle}{TS1}{52}
883 \DeclareTextSymbol{\textfiveoldstyle}{TS1}{53}
884 \DeclareTextSymbol{\textsixoldstyle}{TS1}{54}
885 \DeclareTextSymbol{\textsevenoldstyle}{TS1}{55}

'38 = 56

886 \DeclareTextSymbol{\texteightoldstyle}{TS1}{56}
887 \DeclareTextSymbol{\textnineoldstyle}{TS1}{57}

More text companion symbols.

888 \DeclareTextSymbol{\textlangel}{TS1}{60}
889 \DeclareTextSymbol{\textminus}{TS1}{61}
890 \DeclareTextSymbol{\textrangle}{TS1}{62}

'48 = 72

891 \DeclareTextSymbol{\textmho}{TS1}{77}

The big circle is here to define the command \textcircled. Formerly it was taken from the cmsy font.

892 \DeclareTextSymbol{\textbigcircle}{TS1}{79}
893 \DeclareTextCommand{\textcircled}{TS1}[1]{\hmode@bgroup
894 \oalign{%
895 \hfil \raise .07ex\hbox {\upshape#1}\hfil \crcr
896 \char 79 % '117 = "4F
897 }%
898 \egroup}

More text companion symbols.

'50 = 80

899 \DeclareTextSymbol{\textohm}{TS1}{87}

'58 = 88

900 \DeclareTextSymbol{\textlbrackdbl}{TS1}{91}
901 \DeclareTextSymbol{\textrbrackdbl}{TS1}{93}
902 \DeclareTextSymbol{\textuparrowarrow}{TS1}{94}
903 \DeclareTextSymbol{\textdownarrowarrow}{TS1}{95}

```

```

"60 = 96
904 \DeclareTextSymbol{\textasciigrave}{TS1}{96}
905 \DeclareTextSymbol{\textborn}{TS1}{98}
906 \DeclareTextSymbol{\textdivorced}{TS1}{99}
907 \DeclareTextSymbol{\textdied}{TS1}{100}

"68 = 104
908 \DeclareTextSymbol{\textleaf}{TS1}{108}
909 \DeclareTextSymbol{\textmarried}{TS1}{109}
910 \DeclareTextSymbol{\textmusicalnote}{TS1}{110}

"78 = 120
911 \DeclareTextSymbol{\texttildelow}{TS1}{126}
This glyph, \textdblhyphenchar is hanging, like the hyphenchar of the ec fonts.
912 \DeclareTextSymbol{\textdblhyphenchar}{TS1}{127}

"80 = 128
913 \DeclareTextSymbol{\textasciibreve}{TS1}{128}
914 \DeclareTextSymbol{\textasciicaron}{TS1}{129}
This next glyph is not the same as \textquotedbl.
915 \DeclareTextSymbol{\textacutedbl}{TS1}{130}
916 \DeclareTextSymbol{\textgravedbl}{TS1}{131}
917 \DeclareTextSymbol{\textdagger}{TS1}{132}
918 \DeclareTextSymbol{\textdaggerdbl}{TS1}{133}
919 \DeclareTextSymbol{\textbardbl}{TS1}{134}
920 \DeclareTextSymbol{\textperthousand}{TS1}{135}

"88 = 136
921 \DeclareTextSymbol{\textbullet}{TS1}{136}
922 \DeclareTextSymbol{\textcelsius}{TS1}{137}
923 \DeclareTextSymbol{\textdollaroldstyle}{TS1}{138}
924 \DeclareTextSymbol{\textcentoldstyle}{TS1}{139}
925 \DeclareTextSymbol{\textflorin}{TS1}{140}
926 \DeclareTextSymbol{\textcolonmonetary}{TS1}{141}
927 \DeclareTextSymbol{\textwon}{TS1}{142}
928 \DeclareTextSymbol{\textnaira}{TS1}{143}

"90 = 144
929 \DeclareTextSymbol{\textguarani}{TS1}{144}
930 \DeclareTextSymbol{\textpeso}{TS1}{145}
931 \DeclareTextSymbol{\textlira}{TS1}{146}
932 \DeclareTextSymbol{\textrecipe}{TS1}{147}
933 \DeclareTextSymbol{\textinterrobang}{TS1}{148}
934 \DeclareTextSymbol{\textinterrobangdown}{TS1}{149}
935 \DeclareTextSymbol{\textdong}{TS1}{150}
936 \DeclareTextSymbol{\texttrademark}{TS1}{151}

"98 = 152
937 \DeclareTextSymbol{\textpertenthousand}{TS1}{152}
938 \DeclareTextSymbol{\textpilcrow}{TS1}{153}
939 \DeclareTextSymbol{\textbaht}{TS1}{154}
940 \DeclareTextSymbol{\textnumero}{TS1}{155}

```

This next name may change. For the following sign we know only a german name, which is abziglich. The meaning is something like “commercial minus”. An ASCII ersatz is ./ (dot slash dot). The temporary English name is \textdiscount.

```

941 \DeclareTextSymbol{\textdiscount}{TS1}{156}
942 \DeclareTextSymbol{\textestimated}{TS1}{157}
943 \DeclareTextSymbol{\textopenbullet}{TS1}{158}
944 \DeclareTextSymbol{\textservicemark}{TS1}{159}

"A0 = 160
945 \DeclareTextSymbol{\textlquill}{TS1}{160}
946 \DeclareTextSymbol{\textrquill}{TS1}{161}
947 \DeclareTextSymbol{\textcent}{TS1}{162}
948 \DeclareTextSymbol{\textsterling}{TS1}{163}
949 \DeclareTextSymbol{\textcurrency}{TS1}{164}
950 \DeclareTextSymbol{\textyen}{TS1}{165}
951 \DeclareTextSymbol{\textbrokenbar}{TS1}{166}
952 \DeclareTextSymbol{\textsection}{TS1}{167}

"A8 = 168
953 \DeclareTextSymbol{\textasciidieresis}{TS1}{168}
954 \DeclareTextSymbol{\textcopyright}{TS1}{169}
955 \DeclareTextSymbol{\textordfeminine}{TS1}{170}
956 \DeclareTextSymbol{\textcopyleft}{TS1}{171}
957 \DeclareTextSymbol{\textlnot}{TS1}{172}

The meaning of the circled-P is “sound recording copyright”.

958 \DeclareTextSymbol{\textcircledP}{TS1}{173}
959 \DeclareTextSymbol{\textregistered}{TS1}{174}
960 \DeclareTextSymbol{\textasciimacron}{TS1}{175}

"B0 = 176
961 \DeclareTextSymbol{\textdegree}{TS1}{176}
962 \DeclareTextSymbol{\textpm}{TS1}{177}
963 \DeclareTextSymbol{\texttwosuperior}{TS1}{178}
964 \DeclareTextSymbol{\textthreesuperior}{TS1}{179}
965 \DeclareTextSymbol{\textasciacute}{TS1}{180}
966 \DeclareTextSymbol{\textmu}{TS1}{181} % micro sign
967 \DeclareTextSymbol{\textparagraph}{TS1}{182}
968 \DeclareTextSymbol{\textperiodcentered}{TS1}{183}

"B8 = 184
969 \DeclareTextSymbol{\textreferencemark}{TS1}{184}
970 \DeclareTextSymbol{\textonesuperior}{TS1}{185}
971 \DeclareTextSymbol{\textordmasculine}{TS1}{186}
972 \DeclareTextSymbol{\textsurd}{TS1}{187}
973 \DeclareTextSymbol{\textonequarter}{TS1}{188}
974 \DeclareTextSymbol{\textonehalf}{TS1}{189}
975 \DeclareTextSymbol{\textthreequarters}{TS1}{190}
976 \DeclareTextSymbol{\texteuro}{TS1}{191}

"E0 = 208
977 \DeclareTextSymbol{\texttimes}{TS1}{214}

"F0 = 240
978 \DeclareTextSymbol{\textdiv}{TS1}{246}
979 </TS1>

```

## 1.11 Definitions for the TU encoding

The TU encoding was originally introduced in the contributed package `fontspec` as a Unicode encoding for XeTeX and LuaTeX.

Normally for these engines, the input consists of Unicode characters encoded in UTF-8. There is therefore little need to use the traditional (ASCII) encoding-specific commands

However, sometimes (e.g. for backwards compatibility) it can be useful to access these Unicode characters via such ASCII-based markup. The commands provided here cover the characters in the T1 and TS1 encodings, but specified in Unicode position. Almost all the command names have been mechanically extracted from the `inputenc` UTF-8 support, which is essentially doing a reverse mapping from UTF-8 data to L<sup>A</sup>T<sub>E</sub>X L<sup>I</sup>C<sub>R</sub> commands.

A few additional names for character which were supported in the original `fontspec` version of this file have also been added, even though they are not currently in the default `inputenc` UTF-8 declarations.

980 <\*TU>

In the base interface the Unicode encoding is always known as TU But we parameterize the encoding name to allow for modelling differences in Unicode support by different fonts.

981 \providecommand\UnicodeEncodingName{TU}

As the Unicode encoding, TU, is only currently available with XeTeX or LuaTeX, we detect these engines first, and make adjustments for the differing font loading syntax. For other engines, we issue a warning then abort this file, switching back to T1 encoding.

```
982 \begingroup\expandafter\expandafter\expandafter\endgroup
983 \expandafter\ifx\csname XeTeXrevision\endcsname\relax

984 \begingroup\expandafter\expandafter\expandafter\endgroup
985 \expandafter\ifx\csname directlua\endcsname\relax
```

Not LuaTeX or XeTeX, abort with a warning.

```
986 \PackageWarningNoLine{fontenc}
987 {\UnicodeEncodingName\space
988 encoding is only available with XeTeX and LuaTeX.\MessageBreak
989 Defaulting to T1 encoding}
990 \def\encodingdefault{T1}
991 \expandafter\expandafter\expandafter\endinput
992 \else
```

LuaTeX. For LuaTeX 1.10+, define a Lua function to disable any handing by the font code. Otherwise we reload the font without TeX ligatures.

```
993 \def\UnicodeFontTeXLigatures{+tlig;}\relax
994 \ifnum\luatexversion<110
995 \def\reserved@a{\%
996 \def\@remove@tlig##1{\@remove@tlig@##1\@nil\@nil\relax}
997 \def\@remove@tlig@##1#1{\@remove@tlig@##1}
998 }\edef\reserved@b{\detokenize{+tlig;}}
999 \expandafter\reserved@a\expandafter{\reserved@b}
1000 \def\@remove@tlig@@##1\@nil#2\relax{#1}
```

```

1001 \def\remove@tlig#1{%
1002 \begingroup
1003 \font\remove@tlig
1004 \expandafter\@remove@tlig\expandafter{\fontname\font}%
1005 \remove@tlig
1006 \char#1\relax
1007 \endgroup
1008 }
1009 \else
1010 \newprotectedluacmd\@remove@tlig@@@%

```

Now we can define the function. Mostly we just have to insert a protected glyph node, which is a glyph node with subtype 256. But we have to keep track of the current mode to avoid inserting the glyph into a vlist.

```

1011 \now@and@everyjob{\directlua{
1012 local rawchar_func = token.create'@remove@tlig@@@'.index
1013 local forcehmode = tex.forcehmode
1014 local put_next = token.put_next
1015 local glyph_id = node.id'glyph'
1016 local rawchar_token = token.new(rawchar_func, token.command_id'lua_call')
1017 lua.get_functions_table()[rawchar_func] = function()
1018 local mode = tex.nest.top.mode
1019 if mode == 1 or mode == -1 then
1020 put_next(rawchar_token)
1021 return forcehmode(true)
1022 end
1023 local n = node.new(glyph_id, 256)
1024 n.font = font.current()
1025 n.char = token.scan_int()
1026 return node.write(n)
1027 end
1028 }}

```

Now `\remove@tlig` can be implemented almost as in XeTeX.

```

1029 \def\remove@tlig#1{\@remove@tlig@@@#1\relax}
1030 \fi
1031 \fi
1032 \else
XeTeX
1033 \def\UnicodeFontTeXLigatures{mapping=tex-text;}
1034 \def\remove@tlig#1{\XeTeXglyph\numexpr\XeTeXcharglyph#1\relax}
1035 \fi
1036 \def\UnicodeFontFile#1#2{"[#1]:#2"}
1037 \def\UnicodeFontName#1#2{"#1:#2"}

 Declare the encoding
1038 \DeclareFontEncoding\UnicodeEncodingName{}{}

 Declare accent command to use a postspended combining character rather than the
TeX \accent primitive
1039 \def\add@unicode@accent#1#2{%
1040 \if\relax\detokenize{#2}\relax^\a0\else#2\fi
1041 \char#1\relax}

```

In its original implementation `\DeclareUnicodeAccent` was given 3 arguments (with second the “Unicode encoding” a.k.a., `\UnicodeEncodingName`) while in other places, e.g., `\DeclareUnicodeComposite`, we always made encoding implicit. So we now change it here to implicit too so that the interfaces become a bit more consistent. To avoid making that a breaking change (even though it only affects two packages on CTAN) we test for #2 being `\UnicodeEncodingName`. This would not catch if somebody used `\DeclareUnicodeAccent{\=}{TU-sub}{“0304”}` but that fortunately hasn’t happened. With the implicit argument you would need to change `\UnicodeEncodingName` instead, as you have to do anyway for the other interface commands.

```

1042 \def\DeclareUnicodeAccent#1#2{%
1043 \edef\reserved@a{#2}%
1044 \edef\reserved@b{\UnicodeEncodingName}%
1045 \ifx\reserved@a\reserved@b
1046 \def\reserved@a{\DeclareUnicodeAccent@{#1}}%
1047 \else
1048 \def\reserved@a{\DeclareUnicodeAccent@{#1}\UnicodeEncodingName}%
1049 \fi
1050 \reserved@a{#2}%
1051 }
1052 \def\DeclareUnicodeAccent@#1#2#3{%
1053 \DeclareTextCommand{#1}{#2}{\addunicode@accent{#3}}%
1054 }
```

Wrapper around `\DeclareTextCompositeCommand` that uses the declared composite if it exists in the current font or falls back to the default definition for the TU accent if not.

```

1055 {
1056 \catcode\z@=11\relax
1057 \gdef\DeclareUnicodeComposite#1#2#3{%
1058 \def\reserved@a##1##2{%
1059 \DeclareTextCompositeCommand#1\UnicodeEncodingName{#2}{%
1060 \iffontchar\font#3 ##2%
1061 \else ##1\fi}%
1062 \expandafter\expandafter\expandafter\extract@default@composite
1063 \csname\UnicodeEncodingName\string#1\endcsname{#2}\@nil
1064 \bgroup
1065 \lccode\z@#3 %
1066 \lowercase{\egroup
1067 \expandafter\reserved@a\expandafter{\reserved@b}{^{\z@}}}}%
1068 }
1069 \def\extract@default@composite#1{%
1070 \ifx\@text@composite#1%
1071 \expandafter\extract@default@composite@a
1072 \else
1073 \expandafter\extract@default@composite@b\expandafter#1%
1074 \fi}
1075 \def\extract@default@composite@a#1\@text@composite#2\@nil{%
1076 \def\reserved@b{#2}}
1077 \def\extract@default@composite@b#1#2\@nil{%
1078 \def\reserved@b{#1#2}}
```

Next two commands are simply syntactic sugar to go with the other `\DeclareUnicode...` declarations.

```

1079 \def\DeclareUnicodeSymbol#1{\DeclareTextSymbol{#1}{\UnicodeEncodingName}}
1080 \def\DeclareUnicodeCommand#1{\DeclareTextCommand{#1}{\UnicodeEncodingName}}
1081 \DeclareUnicodeCommand{textquotesingle} {\remove@tlig{"0027}}
1082 \DeclareUnicodeCommand{textasciigrave} {\remove@tlig{"0060}}
1083 \DeclareUnicodeCommand{textquotedbl} {\remove@tlig{"0022}}
1084 \DeclareUnicodeSymbol{textdollar} {"0024}
1085 \DeclareUnicodeSymbol{textless} {"003C}
1086 \DeclareUnicodeSymbol{textgreater} {"003E}
1087 \DeclareUnicodeSymbol{textbackslash} {"005C}
1088 \DeclareUnicodeSymbol{textasciicircum} {"005E}
1089 \DeclareUnicodeSymbol{textunderscore} {"005F}
1090 \DeclareUnicodeSymbol{textbraceleft} {"007B}
1091 \DeclareUnicodeSymbol{textbar} {"007C}
1092 \DeclareUnicodeSymbol{textbraceright} {"007D}
1093 \DeclareUnicodeSymbol{textasciitilde} {"007E}
1094 \DeclareUnicodeSymbol{textexclamdown} {"00A1}
1095 \DeclareUnicodeSymbol{textcent} {"00A2}
1096 \DeclareUnicodeSymbol{textsterling} {"00A3}
1097 \DeclareUnicodeSymbol{textcurrency} {"00A4}
1098 \DeclareUnicodeSymbol{textyen} {"00A5}
1099 \DeclareUnicodeSymbol{textbrokenbar} {"00A6}
1100 \DeclareUnicodeSymbol{textsection} {"00A7}
1101 \DeclareUnicodeSymbol{textasciidieresis} {"00A8}
1102 \DeclareUnicodeSymbol{textcopyright} {"00A9}
1103 \DeclareUnicodeSymbol{textordfeminine} {"00AA}

1104 \DeclareUnicodeSymbol{guillemetleft} {"00AB}
1105 % old Adobe name
1106 \DeclareUnicodeSymbol{guillemotleft} {"00AB}
1107 \DeclareUnicodeSymbol{textlnot} {"00AC}
1108 \DeclareUnicodeSymbol{textregistered} {"00AE}
1109 \DeclareUnicodeSymbol{textasciimacron} {"00AF}
1110 \DeclareUnicodeSymbol{textdegree} {"00B0}
1111 \DeclareUnicodeSymbol{textpm} {"00B1}
1112 \DeclareUnicodeSymbol{texttwosuperior} {"00B2}
1113 \DeclareUnicodeSymbol{textthreesuperior} {"00B3}
1114 \DeclareUnicodeSymbol{textasciacute} {"00B4}
1115 \DeclareUnicodeSymbol{textmu} {"00B5}
1116 \DeclareUnicodeSymbol{textparagraph} {"00B6}
1117 \DeclareUnicodeSymbol{textperiodcentered} {"00B7}
1118 \DeclareUnicodeSymbol{textonesuperior} {"00B9}
1119 \DeclareUnicodeSymbol{textordmasculine} {"00BA}

1120 \DeclareUnicodeSymbol{guillemetright} {"00BB}
1121 % old Adobe name
1122 \DeclareUnicodeSymbol{guillemotright} {"00BB}
1123 \DeclareUnicodeSymbol{textonequarter} {"00BC}
1124 \DeclareUnicodeSymbol{textonehalf} {"00BD}
1125 \DeclareUnicodeSymbol{textthreequarters} {"00BE}
1126 \DeclareUnicodeSymbol{textquestiondown} {"00BF}
1127 \DeclareUnicodeSymbol{AE} {"00C6}
1128 \DeclareUnicodeSymbol{DH} {"00D0}
1129 \DeclareUnicodeSymbol{texttimes} {"00D7}

```

```

1130 \DeclareUnicodeSymbol{\O} {"00D8}
1131 \DeclareUnicodeSymbol{\TH} {"00DE}
1132 \DeclareUnicodeSymbol{\ss} {"00DF}
1133 \DeclareUnicodeSymbol{\ae} {"00E6}
1134 \DeclareUnicodeSymbol{\dh} {"00F0}
1135 \DeclareUnicodeSymbol{\textdiv} {"00F7}
1136 \DeclareUnicodeSymbol{\o} {"00F8}
1137 \DeclareUnicodeSymbol{\th} {"00FE}
1138 \DeclareUnicodeSymbol{\DJ} {"0110}
1139 \DeclareUnicodeSymbol{\dj} {"0111}
1140 \DeclareUnicodeSymbol{\i} {"0131}
1141 \DeclareUnicodeSymbol{\IJ} {"0132}
1142 \DeclareUnicodeSymbol{\ij} {"0133}
1143 \DeclareUnicodeSymbol{\L} {"0141}
1144 \DeclareUnicodeSymbol{\l} {"0142}
1145 \DeclareUnicodeSymbol{\NG} {"014A}
1146 \DeclareUnicodeSymbol{\ng} {"014B}
1147 \DeclareUnicodeSymbol{\OE} {"0152}
1148 \DeclareUnicodeSymbol{\oe} {"0153}
1149 \DeclareUnicodeSymbol{\textflorin} {"0192}
1150 \DeclareUnicodeSymbol{\j} {"0237}
1151 \DeclareUnicodeSymbol{\textasciicaron} {"02C7}
1152 \DeclareUnicodeSymbol{\textasciibreve} {"02D8}
1153 \DeclareUnicodeSymbol{\textacutedbl} {"02DD}
1154 \DeclareUnicodeSymbol{\textgravedbl} {"02F5}
1155 \DeclareUnicodeSymbol{\texttildelow} {"02F7}
1156 \DeclareUnicodeSymbol{\textbaht} {"0E3F}
1157 \DeclareUnicodeSymbol{\SS} {"1E9E}
1158 \DeclareUnicodeSymbol{\textcompwordmark} {"200C}

1159 %\DeclareUnicodeSymbol{\textnonbreakinghyphen} {"2011}
1160 %\DeclareUnicodeSymbol{\textfiguredash} {"2012}
1161 \DeclareUnicodeSymbol{\textendash} {"2013}
1162 \DeclareUnicodeSymbol{\textemdash} {"2014}
1163 %\DeclareUnicodeSymbol{\texthorizontalbar} {"2015}

```

Unfortunately some fonts do not implement "2011, "2012 and/or "2015 (including the L<sup>A</sup>T<sub>E</sub>X default fonts for Unicode engines) so we provide some approximations if the glyph is missing, like we do for OT1 and T1.

The `\nobreak\hspace{z@}` is there to prevent a break after the hyphen but allow later breaks in the remainder of the word.

```

1164 \DeclareUnicodeCommand{\textnonbreakinghyphen}
1165 {\iffontchar\font "2011 \char "2011 \else \mbox{-}\nobreak\hspace{z@} \fi}
1166 \DeclareUnicodeCommand{\textfiguredash}
1167 {\iffontchar\font "2012 \char "2012 \else \char "2013 \fi}
1168 \DeclareUnicodeCommand{\texthorizontalbar}
1169 {\iffontchar\font "2015 \char "2015 \else \char "2014 \fi}

1170 \DeclareUnicodeSymbol{\textbardbl} {"2016}
1171 \DeclareUnicodeSymbol{\textquotel} {"2018}
1172 \DeclareUnicodeSymbol{\textquoter} {"2019}
1173 \DeclareUnicodeSymbol{\quotesinglbase} {"201A}
1174 \DeclareUnicodeSymbol{\textquotedblleft} {"201C}
1175 \DeclareUnicodeSymbol{\textquotedblright} {"201D}
1176 \DeclareUnicodeSymbol{\quotedblbase} {"201E}

```

```

1177 \DeclareUnicodeSymbol{\textdagger} {"2020}
1178 \DeclareUnicodeSymbol{\textdaggerdbl} {"2021}
1179 \DeclareUnicodeSymbol{\textbullet} {"2022}
1180 \DeclareUnicodeSymbol{\textellipsis} {"2026}
1181 \DeclareUnicodeSymbol{\textperthousand} {"2030}
1182 \DeclareUnicodeSymbol{\textpertenthousand} {"2031}
1183 \DeclareUnicodeSymbol{\guilsinglleft} {"2039}
1184 \DeclareUnicodeSymbol{\guilsinglright} {"203A}
1185 \DeclareUnicodeSymbol{\textreferencemark} {"203B}
1186 \DeclareUnicodeSymbol{\textinterrobang} {"203D}
1187 \DeclareUnicodeSymbol{\textfractionsolidus} {"2044}
1188 \DeclareUnicodeSymbol{\textlquill} {"2045}
1189 \DeclareUnicodeSymbol{\textrquill} {"2046}
1190 \DeclareUnicodeSymbol{\textdiscount} {"2052}
1191 \DeclareUnicodeSymbol{\textcolonmonetary} {"20A1}
1192 \DeclareUnicodeSymbol{\textlira} {"20A4}
1193 \DeclareUnicodeSymbol{\textnaira} {"20A6}
1194 \DeclareUnicodeSymbol{\textwon} {"20A9}
1195 \DeclareUnicodeSymbol{\textdong} {"20AB}
1196 \DeclareUnicodeSymbol{\texteuro} {"20AC}
1197 \DeclareUnicodeSymbol{\textpeso} {"20B1}
1198 \DeclareUnicodeSymbol{\textcelsius} {"2103}
1199 \DeclareUnicodeSymbol{\textnumero} {"2116}
1200 \DeclareUnicodeSymbol{\textcircledP} {"2117}
1201 \DeclareUnicodeSymbol{\textrecipie} {"211E}
1202 \DeclareUnicodeSymbol{\textservicemark} {"2120}
1203 \DeclareUnicodeSymbol{\texttrademark} {"2122}
1204 \DeclareUnicodeSymbol{\textohm} {"2126}
1205 \DeclareUnicodeSymbol{\textmho} {"2127}
1206 \DeclareUnicodeSymbol{\textestimated} {"212E}
1207 \DeclareUnicodeSymbol{\textleftarrow} {"2190}
1208 \DeclareUnicodeSymbol{\textuparrow} {"2191}
1209 \DeclareUnicodeSymbol{\textrightarrow} {"2192}
1210 \DeclareUnicodeSymbol{\textdownarrow} {"2193}
1211 \DeclareUnicodeSymbol{\textminus} {"2212}
1212
1213 \DeclareUnicodeSymbol{\Hwithstroke} {"0126}
1214 \DeclareUnicodeSymbol{\hwithstroke} {"0127}

```

Not all fonts have U+2217 but using U+002A requires some adjustment.

```

1215 \DeclareUnicodeCommand{\textasteriskcentered}{%
1216 \iffontchar\font"2217 \char"2217 \else
1217 \begingroup
1218 \fontsize
1219 {\the\dimexpr1.3\dimexpr\f@size pt\relax}%
1220 {\f@baselineskip}%
1221 \selectfont
1222 \raisebox{-0.7ex}{[\dimexpr\height-0.7ex] [0pt]{*}}%
1223 \endgroup
1224 \fi
1225 }
1226 \DeclareUnicodeSymbol{\textsurd} {"221A}
1227 \DeclareUnicodeSymbol{\textlangle} {"2329}

```

```

1228 \DeclareUnicodeSymbol{\textrangle} {"232A}
1229 \DeclareUnicodeSymbol{\textblank} {"2422}
1230 \DeclareUnicodeSymbol{\textvisiblespace} {"2423}
1231 \DeclareUnicodeSymbol{\textopenbullet} {"25E6}
1232 \DeclareUnicodeSymbol{\textbigcircle} {"25EF}
1233 \DeclareUnicodeSymbol{\textmusicalnote} {"266A}
1234 \DeclareUnicodeSymbol{\textmarried} {"26AD}
1235 \DeclareUnicodeSymbol{\textdivorced} {"26AE}
1236 \DeclareUnicodeSymbol{\textinterrobangdown} {"2E18}

```

Accents must be declared before the composites that use them.

```

1237 \DeclareUnicodeAccent{\`}{0300}
1238 \DeclareUnicodeAccent{\'}{0301}
1239 \DeclareUnicodeAccent{\^}{0302}
1240 \DeclareUnicodeAccent{\~}{0303}
1241 \DeclareUnicodeAccent{\=}{0304}
1242 \DeclareUnicodeAccent{\u}{0306}
1243 \DeclareUnicodeAccent{\.}{0307}
1244 \DeclareUnicodeAccent{\"}{0308}
1245 \DeclareUnicodeAccent{\r}{030A}
1246 \DeclareUnicodeAccent{\H}{030B}
1247 \DeclareUnicodeAccent{\v}{030C}
1248 \DeclareUnicodeAccent{\b}{0332}
1249 \DeclareUnicodeAccent{\d}{0323}
1250 \DeclareUnicodeAccent{\c}{0327}
1251 \DeclareUnicodeAccent{\k}{0328}

```

The odd one out:

```

1252 \DeclareUnicodeCommand{textcommabelow[1]
1253 {\hmode@bgroup\oalign{\null#1\crcr\hidewidth\raise-.31ex
1254 \hbox{\check@mathfonts\fontsize\ssf@size\z@
1255 \math@fontsfalse\selectfont,\}\hidewidth}\egroup}
1256 \DeclareUnicodeComposite{\`} {"005E}
1257 \DeclareUnicodeComposite{\~} {"007E}
1258 \DeclareUnicodeComposite{\'} {A>{"00C0}
1259 \DeclareUnicodeComposite{\'} {A>{"00C1}
1260 \DeclareUnicodeComposite{\^} {A>{"00C2}
1261 \DeclareUnicodeComposite{\~} {A>{"00C3}
1262 \DeclareUnicodeComposite{\"} {A>{"00C4}
1263 \DeclareUnicodeComposite{\r} {A>{"00C5}
1264 \DeclareUnicodeComposite{\c} {C>{"00C7}
1265 \DeclareUnicodeComposite{\'} {E>{"00C8}
1266 \DeclareUnicodeComposite{\'} {E>{"00C9}
1267 \DeclareUnicodeComposite{\^} {E>{"00CA}
1268 \DeclareUnicodeComposite{\"} {E>{"00CB}
1269 \DeclareUnicodeComposite{\'} {I>{"00CC}
1270 \DeclareUnicodeComposite{\'} {I>{"00CD}
1271 \DeclareUnicodeComposite{\~} {I>{"00CE}
1272 \DeclareUnicodeComposite{\"} {I>{"00CF}
1273 \DeclareUnicodeComposite{\~} {N>{"00D1}
1274 \DeclareUnicodeComposite{\'} {O>{"00D2}
1275 \DeclareUnicodeComposite{\'} {O>{"00D3}
1276 \DeclareUnicodeComposite{\^} {O>{"00D4}
1277 \DeclareUnicodeComposite{\~} {O>{"00D5}

```

```

1278 \DeclareUnicodeComposite{\"}
1279 \DeclareUnicodeComposite{\'}
1280 \DeclareUnicodeComposite{\`}
1281 \DeclareUnicodeComposite{\`}
1282 \DeclareUnicodeComposite{\"}
1283 \DeclareUnicodeComposite{\`}
1284 \DeclareUnicodeComposite{\'}
1285 \DeclareUnicodeComposite{\`}
1286 \DeclareUnicodeComposite{\`}
1287 \DeclareUnicodeComposite{\`}
1288 \DeclareUnicodeComposite{\"}
1289 \DeclareUnicodeComposite{\r}
1290 \DeclareUnicodeComposite{\c}
1291 \DeclareUnicodeComposite{\`}
1292 \DeclareUnicodeComposite{\`}
1293 \DeclareUnicodeComposite{\`}
1294 \DeclareUnicodeComposite{\"}
1295 \DeclareUnicodeComposite{\'}
1296 \DeclareUnicodeComposite{\'}
1297 \DeclareUnicodeComposite{\`}
1298 \DeclareUnicodeComposite{\`}
1299 \DeclareUnicodeComposite{\`}
1300 \DeclareUnicodeComposite{\`}
1301 \DeclareUnicodeComposite{\"}
1302 \DeclareUnicodeComposite{\"}
1303 \DeclareUnicodeComposite{\`}
1304 \DeclareUnicodeComposite{\`}
1305 \DeclareUnicodeComposite{\`}
1306 \DeclareUnicodeComposite{\`}
1307 \DeclareUnicodeComposite{\`}
1308 \DeclareUnicodeComposite{\"}
1309 \DeclareUnicodeComposite{\`}
1310 \DeclareUnicodeComposite{\`}
1311 \DeclareUnicodeComposite{\`}
1312 \DeclareUnicodeComposite{\"}
1313 \DeclareUnicodeComposite{\`}
1314 \DeclareUnicodeComposite{\"}
1315 \DeclareUnicodeComposite{\=}
1316 \DeclareUnicodeComposite{\=}
1317 \DeclareUnicodeComposite{\u}
1318 \DeclareUnicodeComposite{\u}
1319 \DeclareUnicodeComposite{\k}
1320 \DeclareUnicodeComposite{\k}
1321 \DeclareUnicodeComposite{\`}
1322 \DeclareUnicodeComposite{\`}
1323 \DeclareUnicodeComposite{\`}
1324 \DeclareUnicodeComposite{\`}
1325 \DeclareUnicodeComposite{\.}
1326 \DeclareUnicodeComposite{\.}
1327 \DeclareUnicodeComposite{\v}
1328 \DeclareUnicodeComposite{\v}
1329 \DeclareUnicodeComposite{\v}
1330 \DeclareUnicodeComposite{\v}
1331 \DeclareUnicodeComposite{\=}
{O}{"00D6}
{U}{"00D9}
{U}{"00DA}
{U}{"00DB}
{U}{"00DC}
{Y}{"00DD}
{a}{"00E0}
{a}{"00E1}
{a}{"00E2}
{a}{"00E3}
{a}{"00E4}
{a}{"00E5}
{c}{"00E7}
{e}{"00E8}
{e}{"00E9}
{e}{"00EA}
{e}{"00EB}
\i {"00EC}
{i}{"00EC}
\i {"00ED}
{i}{"00ED}
\i {"00EE}
{i}{"00EE}
\i {"00EF}
{i}{"00EF}
{n}{"00F1}
{o}{"00F2}
{o}{"00F3}
{o}{"00F4}
{o}{"00F5}
{o}{"00F6}
{u}{"00F9}
{u}{"00FA}
{u}{"00FB}
{u}{"00FC}
{y}{"00FD}
{y}{"00FF}
{A}{"0100}
{a}{"0101}
{A}{"0102}
{a}{"0103}
{A}{"0104}
{a}{"0105}
{C}{"0106}
{c}{"0107}
{C}{"0108}
{c}{"0109}
{C}{"010A}
{c}{"010B}
{C}{"010C}
{c}{"010D}
{D}{"010E}
{d}{"010F}
{E}{"0112}

```

```

1332 \DeclareUnicodeComposite{\=}{e}{0113}
1333 \DeclareUnicodeComposite{\u}{E}{0114}
1334 \DeclareUnicodeComposite{\u}{e}{0115}
1335 \DeclareUnicodeComposite{\.}{E}{0116}
1336 \DeclareUnicodeComposite{\.}{e}{0117}
1337 \DeclareUnicodeComposite{\k}{E}{0118}
1338 \DeclareUnicodeComposite{\k}{e}{0119}
1339 \DeclareUnicodeComposite{\v}{E}{011A}
1340 \DeclareUnicodeComposite{\v}{e}{011B}
1341 \DeclareUnicodeComposite{\^}{G}{011C}
1342 \DeclareUnicodeComposite{\^}{g}{011D}
1343 \DeclareUnicodeComposite{\u}{G}{011E}
1344 \DeclareUnicodeComposite{\u}{g}{011F}
1345 \DeclareUnicodeComposite{\.}{G}{0120}
1346 \DeclareUnicodeComposite{\.}{g}{0121}
1347 \DeclareUnicodeComposite{\c}{G}{0122}
1348 \DeclareUnicodeComposite{\c}{g}{0123}
1349 \DeclareUnicodeComposite{\^}{H}{0124}
1350 \DeclareUnicodeComposite{\^}{h}{0125}
1351 \DeclareUnicodeComposite{\~}{I}{0128}
1352 \DeclareUnicodeComposite{\~}{\i}{0129}
1353 \DeclareUnicodeComposite{\~}{i}{0129}
1354 \DeclareUnicodeComposite{\=}{I}{012A}
1355 \DeclareUnicodeComposite{\=}{\i}{012B}
1356 \DeclareUnicodeComposite{\=}{i}{012B}
1357 \DeclareUnicodeComposite{\u}{I}{012C}
1358 \DeclareUnicodeComposite{\u}{\i}{012D}
1359 \DeclareUnicodeComposite{\u}{i}{012D}
1360 \DeclareUnicodeComposite{\k}{I}{012E}
1361 \DeclareUnicodeComposite{\k}{\i}{012F}
1362 \DeclareUnicodeComposite{\k}{i}{012F}
1363 \DeclareUnicodeComposite{\.}{I}{0130}
1364 \DeclareUnicodeComposite{\^}{J}{0134}
1365 \DeclareUnicodeComposite{\^}{j}{0135}
1366 \DeclareUnicodeComposite{\^}{j}{0135}
1367 \DeclareUnicodeComposite{\c}{K}{0136}
1368 \DeclareUnicodeComposite{\c}{k}{0137}
1369 \DeclareUnicodeComposite{\'}{L}{0139}
1370 \DeclareUnicodeComposite{\'}{l}{013A}
1371 \DeclareUnicodeComposite{\c}{L}{013B}
1372 \DeclareUnicodeComposite{\c}{l}{013C}
1373 \DeclareUnicodeComposite{\v}{L}{013D}
1374 \DeclareUnicodeComposite{\v}{l}{013E}
1375 \DeclareUnicodeComposite{\'}{N}{0143}
1376 \DeclareUnicodeComposite{\'}{n}{0144}
1377 \DeclareUnicodeComposite{\c}{N}{0145}
1378 \DeclareUnicodeComposite{\c}{n}{0146}
1379 \DeclareUnicodeComposite{\v}{N}{0147}
1380 \DeclareUnicodeComposite{\v}{n}{0148}
1381 \DeclareUnicodeComposite{\=}{O}{014C}
1382 \DeclareUnicodeComposite{\=}{o}{014D}
1383 \DeclareUnicodeComposite{\u}{O}{014E}
1384 \DeclareUnicodeComposite{\u}{o}{014F}
1385 \DeclareUnicodeComposite{\H}{O}{0150}

```

```

1386 \DeclareUnicodeComposite{\H} {o}{0151}
1387 \DeclareUnicodeComposite{\'} {R}{0154}
1388 \DeclareUnicodeComposite{\'} {r}{0155}
1389 \DeclareUnicodeComposite{\c} {R}{0156}
1390 \DeclareUnicodeComposite{\c} {r}{0157}
1391 \DeclareUnicodeComposite{\v} {R}{0158}
1392 \DeclareUnicodeComposite{\v} {r}{0159}
1393 \DeclareUnicodeComposite{\'} {S}{015A}
1394 \DeclareUnicodeComposite{\'} {s}{015B}
1395 \DeclareUnicodeComposite{\^} {S}{015C}
1396 \DeclareUnicodeComposite{\^} {s}{015D}
1397 \DeclareUnicodeComposite{\c} {S}{015E}
1398 \DeclareUnicodeComposite{\c} {s}{015F}
1399 \DeclareUnicodeComposite{\v} {S}{0160}
1400 \DeclareUnicodeComposite{\v} {s}{0161}
1401 \DeclareUnicodeComposite{\c} {T}{0162}
1402 \DeclareUnicodeComposite{\c} {t}{0163}
1403 \DeclareUnicodeComposite{\v} {T}{0164}
1404 \DeclareUnicodeComposite{\v} {t}{0165}
1405 \DeclareUnicodeComposite{\~} {U}{0168}
1406 \DeclareUnicodeComposite{\~} {u}{0169}
1407 \DeclareUnicodeComposite{\=} {U}{016A}
1408 \DeclareUnicodeComposite{\=} {u}{016B}
1409 \DeclareUnicodeComposite{\u} {U}{016C}
1410 \DeclareUnicodeComposite{\u} {u}{016D}
1411 \DeclareUnicodeComposite{\r} {U}{016E}
1412 \DeclareUnicodeComposite{\r} {u}{016F}
1413 \DeclareUnicodeComposite{\H} {U}{0170}
1414 \DeclareUnicodeComposite{\H} {u}{0171}
1415 \DeclareUnicodeComposite{\k} {U}{0172}
1416 \DeclareUnicodeComposite{\k} {u}{0173}
1417 \DeclareUnicodeComposite{\^} {W}{0174}
1418 \DeclareUnicodeComposite{\^} {w}{0175}
1419 \DeclareUnicodeComposite{\^} {Y}{0176}
1420 \DeclareUnicodeComposite{\^} {y}{0177}
1421 \DeclareUnicodeComposite{\"} {Y}{0178}
1422 \DeclareUnicodeComposite{\'} {Z}{0179}
1423 \DeclareUnicodeComposite{\'} {z}{017A}
1424 \DeclareUnicodeComposite{\.} {Z}{017B}
1425 \DeclareUnicodeComposite{\.} {z}{017C}
1426 \DeclareUnicodeComposite{\v} {Z}{017D}
1427 \DeclareUnicodeComposite{\v} {z}{017E}
1428 \DeclareUnicodeComposite{\v} {A}{01CD}
1429 \DeclareUnicodeComposite{\v} {a}{01CE}
1430 \DeclareUnicodeComposite{\v} {I}{01CF}
1431 \DeclareUnicodeComposite{\v} {i}{01D0}
1432 \DeclareUnicodeComposite{\v} {i}{01D0}
1433 \DeclareUnicodeComposite{\v} {O}{01D1}
1434 \DeclareUnicodeComposite{\v} {o}{01D2}
1435 \DeclareUnicodeComposite{\v} {U}{01D3}
1436 \DeclareUnicodeComposite{\v} {u}{01D4}

1437 \DeclareUnicodeComposite{\'} {\AE}{01FC}
1438 \DeclareUnicodeComposite{\'} {\xE}{01FC}

```

```

1439 \DeclareUnicodeComposite{\'} \ae{"01FD}
1440 \DeclareUnicodeComposite{\'} {\æ}{"01FD}
1441 \DeclareUnicodeComposite{\=} \AE{"01E2}
1442 \DeclareUnicodeComposite{\=} {\Ѐ}{"01E2}
1443 \DeclareUnicodeComposite{\=} \ae{"01E3}
1444 \DeclareUnicodeComposite{\=} {\ѧ}{"01E3}
1445 \DeclareUnicodeComposite{\v} \G{"01E6}
1446 \DeclareUnicodeComposite{\v} \g{"01E7}
1447 \DeclareUnicodeComposite{\v} \K{"01E8}
1448 \DeclareUnicodeComposite{\v} \k{"01E9}
1449 \DeclareUnicodeComposite{\k} \O{"01EA}
1450 \DeclareUnicodeComposite{\k} \o{"01EB}
1451 \DeclareUnicodeComposite{\v} \j {"01FO}
1452 \DeclareUnicodeComposite{\v} \j {"01FO}
1453 \DeclareUnicodeComposite{\'} \G{"01F4}
1454 \DeclareUnicodeComposite{\'} \g{"01F5}
1455 \DeclareUnicodeComposite{\textcommabelow}{S}{"0218}
1456 \DeclareUnicodeComposite{\textcommabelow}{s}{"0219}
1457 \DeclareUnicodeComposite{\textcommabelow}{T}{"021A}
1458 \DeclareUnicodeComposite{\textcommabelow}{t}{"021B}
1459 \DeclareUnicodeComposite{\=} \Y{"0232}
1460 \DeclareUnicodeComposite{\=} \y{"0233}
1461 \DeclareUnicodeComposite{\.} \B{"1E02}
1462 \DeclareUnicodeComposite{\.} \b{"1E03}
1463 \DeclareUnicodeComposite{\d} \B{"1E04}
1464 \DeclareUnicodeComposite{\d} \b{"1E05}
1465 \DeclareUnicodeComposite{\d} \D{"1E0C}
1466 \DeclareUnicodeComposite{\d} \d{"1E0D}
1467 \DeclareUnicodeComposite{\=} \G{"1E20}
1468 \DeclareUnicodeComposite{\=} \g{"1E21}
1469 \DeclareUnicodeComposite{\d} \H{"1E24}
1470 \DeclareUnicodeComposite{\d} \h{"1E25}
1471 \DeclareUnicodeComposite{\d} \K{"1E32}
1472 \DeclareUnicodeComposite{\d} \k{"1E33}
1473 \DeclareUnicodeComposite{\d} \L{"1E36}
1474 \DeclareUnicodeComposite{\d} \l {"1E37}
1475 \DeclareUnicodeComposite{\d} \M{"1E42}
1476 \DeclareUnicodeComposite{\d} \m {"1E43}
1477 \DeclareUnicodeComposite{\d} \N {"1E46}
1478 \DeclareUnicodeComposite{\d} \n {"1E47}
1479 \DeclareUnicodeComposite{\d} \R {"1E5A}
1480 \DeclareUnicodeComposite{\d} \r {"1E5B}
1481 \DeclareUnicodeComposite{\d} \S {"1E62}
1482 \DeclareUnicodeComposite{\d} \s {"1E63}
1483 \DeclareUnicodeComposite{\d} \T {"1E6C}
1484 \DeclareUnicodeComposite{\d} \t {"1E6D}
1485 \DeclareUnicodeComposite{\d} \V {"1E7E}
1486 \DeclareUnicodeComposite{\d} \v {"1E7F}
1487 \DeclareUnicodeComposite{\d} \W {"1E88}
1488 \DeclareUnicodeComposite{\d} \w {"1E89}
1489 \DeclareUnicodeComposite{\d} \Z {"1E92}
1490 \DeclareUnicodeComposite{\d} \z {"1E93}
1491 \DeclareUnicodeComposite{\d} \A {"1EA0}
1492 \DeclareUnicodeComposite{\d} \a {"1EA1}

```

```

1493 \DeclareUnicodeComposite{\d} {E}{1EB8}
1494 \DeclareUnicodeComposite{\d} {e}{1EB9}
1495 \DeclareUnicodeComposite{\d} {I}{1ECA}
1496 \DeclareUnicodeComposite{\d} {i}{1ECB}
1497 \DeclareUnicodeComposite{\d} {O}{1ECC}
1498 \DeclareUnicodeComposite{\d} {o}{1ECD}
1499 \DeclareUnicodeComposite{\d} {U}{1EE4}
1500 \DeclareUnicodeComposite{\d} {u}{1EE5}
1501 \DeclareUnicodeComposite{\d} {Y}{1EF4}
1502 \DeclareUnicodeComposite{\d} {y}{1EF5}

1503
```

## 2 Package files

This file now also contains some packages that provide access to the more specialised encodings.

### 2.1 The fontenc package

This package allows authors to specify which encodings they will use. For each encoding `FOO`, the package looks to see if the encoding `FOO` has already been declared. If it has not, the file `fooenc.def` is loaded. The default encoding is set to be `FOO`.

In addition the package at the moment contains extra code to extend the `\@uclclist` (list of upper/lower case pairs) for encodings that involve cyrillic characters. THIS IS A TEMPORARY SOLUTION and will not stay this way forever (or so we hope) but right now we are missing a proper interface for this and didn't wanted to rush it.

```
1504 <*package>
```

Here we define a macro that extends the `\@uclclist` if needed and afterwards turns itself in a noop.

```

1505 \def\update@uclc@with@cyrilllic{%
1506 \expandafter\def\expandafter@\uclclist\expandafter
1507 {\@uclclist
1508 \cyla\CYRA\cyrabch\CYRABHCH\cyrabchdsc\CYRABHCHDSC\cyrabhdze
1509 \CYRABHDZE\cyrabha\CYRABHA\cyrae\CYRAE\cyrb\CYRB\cyrbyus
1510 \CYRBYUS\cyc\CYRC\cyrch\CYRCH\cyrchdsc\CYRCHLDSC\cyrchrdsc
1511 \CYRCHRDS\cyrchvcrs\CYRCHVCRS\cyrd\CYRD\cyrdelta\CYRDELTA
1512 \cyrdje\CYRDJE\cyrdze\CYRDZE\cyrdzhe\CYRDZHE\cyre\CYRE\cyreps
1513 \CYREPS\cyrerev\CYREREV\cyrery\CYRERY\cyrf\CYRF\cyrfita
1514 \CYRFITA\cyrg\CYRG\cyrgdsc\CYRGDSC\cyrgdschcrs\CYRGDSCHCRS
1515 \cyrghcrs\CYRGHCRS\cyrghk\CYRGHK\cyrgup\CYRGUP\cyrh\CYRH
1516 \cyrhdsc\CYRHDSC\cyrhhcrs\CYRHHCRS\cyrhhk\CYRHHK\cyrhdsn
1517 \CYRHRDSN\cyri\CYRI\cyrie\CYRIE\cyrii\CYRII\cyrishrt\CYRISHRT
1518 \cyrishrtsc\CYRISHRTDSC\cyrizh\CYRIZH\cyrje\CYRJE\cyrk\CYRK
1519 \cyrkbeak\CYRKBEAK\cyrkdsc\CYRKDSC\cyrkhcrs\CYRKHCRS\cyrkhk
1520 \CYRKHK\cyrkvcrs\CYRKVCRS\cyrl\CYRL\cyrlldsc\CYRLDSC\cyrlhk
1521 \CYRLHK\cyrlje\CYRLJE\cyrm\CYRM\cyrmndsc\CYRMNDSC\cyrmhk\CYRMHK
1522 \cyrn\CYRN\cyrndsc\CYRNDSC\cyrng\CYRNG\cyrnhk\CYRNHK\cyrnje
1523 \CYRNJE\cyrnlhk\CYRNLHK\cyro\CYRO\cyrotld\CYROTLD\cyrp\CYRP
1524 \cyrphk\CYRPHK\cyrq\CYRQ\cyrr\CYRR\cyrrdsc\CYRRDSC\cyrrhk
1525 \CYRRHK\cyrrtick\CYRRTICK\cyrs\CYRS\cystsacrs\CYRSACRS
1526 \cystscha\CYRSCHWA\cystsdesc\CYRSDESC\cystssemisftsn\CYRSEMISFTSN

```

```

1527 \cyrsoftsn\CYRSFTSN\cyrsh\CYRSH\cyrshch\CYRSHCH\cyrshha\CYRSHHA
1528 \cyrt\CYRT\cyrtdesc\CYRTDSC\cyrtetse\CYRTETSE\cyrtshe\CYRTSHE
1529 \cyr\CYRU\cyrushrt\CYRUSHRT\cyrv\CYRV\cyrw\CYRW\cqry\CYRY
1530 \carya\CYRYA\cqryat\CYRYAT\cqryhcrs\CYRYHCRS\cqryi\CYRYI\cqryo
1531 \CYRYO\cqryu\CYRYU\cqryz\CYRZ\cqrydsc\CYRZDSC\cqryh\CYRZH
1532 \cqryhdsc\CYRZHDSC}%
1533 \let\update@uclc@with@cyrillic\relax
1534 }

```

Here we process each option:

```

1535 \DeclareOption*{%
1536 \let\encodingdefault\CurrentOption

```

From 2020/02/02 release onward we only load the encoding files if they haven't be loaded already. To check this we look if `\T@encoding` is already defined. If not we load (indicated by setting the switch `@tempswa` to true and we always load if we run in an older format (or rather in a rollback situation).

```

1537 \tempswafalse
1538 \cifl@t@r\fmtversion{2020/02/02}%
1539 {\expandafter\ifx\csname T@\CurrentOption\endcsname\relax
1540 \tempswatrue\fi}%
1541 {\tempswatrue}%

```

Load if necessary:

```

1542 \if@tempswa
1543 \edef\reserved@f{%
1544 \lowercase{\def\noexpand\reserved@f{\CurrentOption enc.def}}%
1545 \reserved@f
1546 \InputIfFileExists\reserved@f
1547 {}{\PackageError{fontenc}%
1548 {Encoding file '\reserved@f' not found.%}
1549 \MessageBreak
1550 You might have misspelt the name of the encoding}%
1551 {Necessary code for this encoding was not
1552 loaded.\MessageBreak
1553 Thus calling the encoding later on will
1554 produce further error messages.}%
1555 \let\reserved@f\relax

```

In case the current encoding is one of a list of known cyrillic ones we extend the `\@uclclist`:

```

1556 \expandafter\in@\expandafter{\CurrentOption}%
1557 {T2A,T2B,T2C,X2,LCY,OT2}%
1558 \ifin@

```

But only if it hasn't already been extended. This might happen if there are several calls to fontenc loading one of the above encodings. If we don't do this check the `\@uclclist` gets unnecessarily big, slowing down the processing at runtime.

```

1559 \expandafter\in@\expandafter\cyra\expandafter
1560 {\@uclclist}%
1561 \ifin@
1562 \else
1563 \update@uclc@with@cyrillic
1564 \fi
1565 \fi

```

```

1566 \fi
1567 }
1568 \ProcessOptions*
```

We select the new font encoding default (i.e., the last encoding specified in the option list. But this encoding may not work with the current `\f@shape`, e.g., LY1 is not defined for `cmr` and therefore packages switching to LY1 usually also change `\rmdefault`. But that only applies at `\begin{document}` so we get a spurious warning if we use what L<sup>A</sup>T<sub>E</sub>X previously used:

```

1569 \%fontencoding\encodingdefault\selectfont
```

So instead we do this here:

```

1570 \usefont\encodingdefault\familydefault\seriesdefault\shapedefault
```

To save some space we get rid of the macro extending the `\@uclclist` (might have happened already).

```

1571 \let\update@uclc@with@cyrillic\relax
```

Finally we pretend that the `fontenc` package wasn't read in. This allows for using it several times, e.g., in a class file and in the preamble (at the cost of not getting any version info). That kind of hackery shows that using a general purpose package just for loading an encoding is not the right kind of interface for setting up encodings — it will get replaced at some point in the future.

```

1572 \let\@elt\relax
1573 \xdef\@fontenc@load@list{\@fontenc@load@list
1574 \@elt{\csname opt@fontenc.sty\endcsname}}
1575 \global\expandafter\let\csname ver@fontenc.sty\endcsname\relax
1576 \global\expandafter\let\csname opt@fontenc.sty\endcsname\relax
1577 \global\let\@ifl@ter@@\@ifl@ter
1578 \def\@ifl@ter#1#2#3#4#5{\global\let\@ifl@ter\@ifl@ter@@}
1579
```

File t

# ltcounts.dtx

## 1 Counters and Lengths

Commands for defining and using counters. This file defines:

|                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\newcounter</code>     | To define a new counter.                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>\setcounter</code>     | To set the value of counters.                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <code>\addtocounter</code>   | Increase the counter #1 by the number #2.                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>\stepcounter</code>    | Increase a counter by one.                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>\refstepcounter</code> | Increase a counter by one, also setting the value used by <code>\label</code> .                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>\value</code>          | For accessing the value of the counter as a TeX number (as opposed to <code>\the&lt;counter&gt;</code> which expands to the <i>printed</i> representation of <code>&lt;counter&gt;</code> )                                                                                                                                                                                                                                                                      |
| <code>\arabic</code>         | <code>\arabic{&lt;counter&gt;}</code> : 1, 2, 3, ...                                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>\roman</code>          | <code>\roman{&lt;counter&gt;}</code> : i, ii, iii, ...                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>\Roman</code>          | <code>\Roman{&lt;counter&gt;}</code> : I, II, III, ...                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>\alph</code>           | <code>\alph{&lt;counter&gt;}</code> : a, b, c, ...                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>\Alpha</code>          | <code>\Alpha{&lt;counter&gt;}</code> : A, B, C, ...                                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>\fnsymbol</code>       | <code>\fnsymbol{&lt;counter&gt;}</code> : *, †, ‡, ...                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>\counterwithin</code>  | <code>\counterwithin[&lt;format&gt;]{&lt;counter&gt;}{&lt;within-counter&gt;}</code> : Resets <code>&lt;counter&gt;</code> whenever <code>&lt;within-counter&gt;</code> is stepped. Also redefines <code>\the&lt;counter&gt;</code> command to produce <code>\the&lt;within-counter&gt;.(&lt;format&gt;){&lt;counter&gt;}</code> with <code>\arabic</code> as the default for <code>&lt;format&gt;</code> . Star form omits redefining the print representation. |
| <code>\counterwithout</code> | <code>\counterwithout[&lt;format&gt;]{&lt;counter&gt;}{&lt;within-counter&gt;}</code> : Removes <code>&lt;counter&gt;</code> from the reset list of <code>&lt;within-counter&gt;</code> . Also redefines <code>\the&lt;counter&gt;</code> command to produce <code>&lt;format&gt;{&lt;counter&gt;}</code> with <code>\arabic</code> as the default for <code>&lt;format&gt;</code> . Star form omits redefining the print representation.                        |
| 1 <code>(*2ekernel)</code>   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

### 1.1 Environment Counter Macros

An environment foo has an associated counter defined by the following control sequences:

|                      |                                                                                                                                                                                                                                                                                                            |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\c@foo</code>  | Contains the counter's numerical value. It is defined by <code>\newcount\foocounter</code> .                                                                                                                                                                                                               |
| <code>\thefoo</code> | Macro that expands to the printed value of <code>\foocounter</code> . For example, if sections are numbered within chapters, and section headings look like                                                                                                                                                |
|                      | Section II-3. The Nature of Counters                                                                                                                                                                                                                                                                       |
|                      | then <code>\thesection</code> might be defined by:                                                                                                                                                                                                                                                         |
|                      | <code>\def\thesection</code><br><code>{\@Roman{\c@chapter}-\@arabic{\c@section}}</code>                                                                                                                                                                                                                    |
| <code>\p@foo</code>  | Macro that expands to a printed 'reference prefix' of counter foo. Any <code>\ref</code> to a value created by counter foo will produce the expansion of <code>\p@foo\thefoo</code> when the <code>\label</code> command is executed. See file <code>ltxref.dtx</code> for an extension of this mechanism. |
| <code>\cl@foo</code> | List of counters to be reset when foo stepped. Has format <code>\@elt{counterA}\@elt{counterB}\@elt{counterC}</code> .                                                                                                                                                                                     |

**NOTE:**

`\thefoo` and `\p@foo` must be defined in such a way that `\edef\bar{\thefoo}` or `\edef\bar{\p@foo}` defines `\bar` so that it will evaluate to the counter value at the time of the `\edef`, even after `\foocounter` and any other counters have been changed. This will happen if you use the standard commands `\@arabic`, `\@Roman`, etc.

The following commands are used to define and modify counters.

`\refstepcounter{<foo>}`

Same as `\stepcounter`, but it also defines `\@currentreference` so that a subsequent `\label{<bar>}` command causes `\ref{<bar>}` to generate the current value of counter `<foo>`.

`\@definecounter{<foo>}`

Initializes counter `{<foo>}` (with empty reset list), defines `\p@foo` and `\thefoo` to be null. Also adds `<foo>` to `\cl@ckpt` – the reset list of a dummy counter `@ckpt` used for taking checkpoints for the `\include` system.

`\@addtoreset{<foo>}{<bar>}` : Adds counter `<foo>` to the list of counters `\cl@bar` to be reset when counter `<bar>` is stepped.

`\@removefromreset{<foo>}{<bar>}` : Removes counter `<foo>` to the list of counters `\cl@bar` to be reset when counter `<bar>` is stepped.

`\setcounter{<foo>}{<val>}` : Globally sets `\foocounter` equal to `<val>`.

```
2 \def\setcounter#1#2{%
3 \@ifundefined{c@#1}%
4 {\@nocounterr{#1}}%
5 {\global\csname c@#1\endcsname#2\relax}}
```

(End definition for `\setcounter`.)

`\addtocounter{<foo>}{<val>}` Globally increments `\foocounter` by `<val>`.

```
6 \def\addtocounter#1#2{%
7 \@ifundefined{c@#1}%
8 {\@nocounterr{#1}}%
9 {\global\advance\csname c@#1\endcsname #2\relax}}
```

(End definition for `\addtocounter`.)

`\newcounter{<newctr>}[<oldctr>]` Defines `<newctr>` to be a counter, which is reset when counter `<oldctr>` is stepped. If `<newctr>` already defined produces ‘`c@newctr already defined`’ error.

```
10 \def\newcounter#1{%
11 \expandafter\@if definable \csname c@#1\endcsname
12 {\@definecounter{#1}}%
13 \@ifnextchar[\{\@newctr{#1}\}{}]
```

(End definition for `\newcounter`.)

`\value{<ctr>}` produces the value of counter `<ctr>`, for use with a `\setcounter` or `\addtocounter` command.

```
14 \def\value#1{\csname c@#1\endcsname}
```

(End definition for `\value`.)

`\@newctr`

```
15 \def\@newctr#1[#2]{%
16 \@if undefined{c@#2}{\@nocounterr{#2}}{\@addtoreset{#1}{#2}}}
```

(End definition for \@newctr.)

\stepcounter \stepcounterfoo Globally increments counter \c@FOO and resets all subsidiary counters.

```
17 \def\stepcounter#1{%
18 \addtocounter{#1}\@ne
19 \begingroup
20 \let\@elt\@stpelt
21 \csname cl@#1\endcsname
22 \endgroup}
```

(End definition for \stepcounter.)

\@stpelt Rather than resetting the “within” counter to zero we set it to -1 and then run \stepcounter that moves it to 0 and also initiates resetting the next level down.

```
23 </2ekernel>
24 <latexrelease>\IncludeInRelease{2015/01/01}{\@stpelt}
25 <latexrelease> {Reset nested counters}%
26 <*2ekernel | latexrelease>
27 \def\@stpelt#1{\global\csname c@#1\endcsname \m@ne\stepcounter{#1}}%
28 <latexrelease>\EndIncludeInRelease
29 </2ekernel | latexrelease>
30 <latexrelease>\IncludeInRelease{0000/00/00}{\@stpelt}
31 <latexrelease> {Reset nested counters}%
32 <latexrelease>\def\@stpelt#1{\global\csname c@#1\endcsname \z@}%
33 <latexrelease>\EndIncludeInRelease
34 <*2ekernel>
```

(End definition for \@stpelt.)

\cl@ckpt

```
35 \def\cl@ckpt{\@elt{page}}
```

(End definition for \cl@ckpt.)

\@definecounter

```
36 \def\@definecounter#1{\expandafter\newcount\csname c@#1\endcsname
37 \setcounter{#1}\z@
38 \global\expandafter\let\csname cl@#1\endcsname\@empty
39 \addtoreset{#1}{\@ckpt}%
40 \global\expandafter\let\csname p@#1\endcsname\@empty
41 \expandafter
42 \gdef\csname the#1\expandafter\endcsname\expandafter
43 {\expandafter\@arabic\csname c@#1\endcsname}}
```

(End definition for \@definecounter.)

\@addtoreset

```
44 \def\@addtoreset#1#2{\expandafter\@cons\csname cl@#2\endcsname {{#1}}}
```

(End definition for \@addtoreset.)

```
45 </2ekernel>
```

```

\@removefromreset
46 <|latexrelease>\IncludeInRelease{2018-04-01}
47 <|latexrelease> {\@removefromreset}{Add interfaces}%
48 (*2ekernel | latexrelease)
49 \def\@removefromreset#1#2{%

```

Even through this is internal and the programmer should know what he/she is doing we test here if counter #2 is defined. If not, the execution would run into a tight loop.

```

50 \@ifundefined{c@#2}\relax
51 {\begingroup
52 \expandafter\let\csname c@#1\endcsname\@removefromreset
53 \def\@elt##1{%
54 \expandafter\ifx\csname c##1\endcsname\@removefromreset
55 \else
56 \noexpand\@elt{##1}%
57 \fi}%
58 \expandafter\xdef\csname cl@#2\endcsname
59 {\csname cl@#2\endcsname}%
60 \endgroup}%

```

*(End definition for \@removefromreset.)*

\@ifbothcounters Test if arg #1 and #2 are counters and if so execute #3.

```

61 \def\@ifbothcounters#1#2#3{%
62 \@ifundefined{c@#1}{\nocounterr{#1}}%
63 {%
64 \@ifundefined{c@#2}{\nocounterr{#2}}%
65 {%
66 \else both counter and within are defined
67 #3}}}

```

*(End definition for \@ifbothcounters.)*

```

68 </2ekernel | latexrelease>
69 <|latexrelease>\EndIncludeInRelease
70 <|latexrelease>\IncludeInRelease{0000-00-00}
71 <|latexrelease> {\@removefromreset}{Add interfaces}%
72 <|latexrelease>\let \@removefromreset \undefined
73 <|latexrelease>\EndIncludeInRelease
74 (*2ekernel)

```

\counterwithout \counterwithin New implementation using xparse and supporting an optional format argument.

```

75 </2ekernel>
76 (*2ekernel | latexrelease)
77 <|latexrelease>\IncludeInRelease{2021/11/15}%
78 <|latexrelease> {\counterwithout}{counter without/within}%
79 \NewDocumentCommand \counterwithout {sO{\arabic}mm}{%
80 \@ifbothcounters{#3}{#4}{%
81 \@removefromreset{#3}{#4}%
82 \IfBooleanF #1{%
83 \expandafter
84 \gdef\csname the#3\endcsname {#2{#3}}{}}%
85 }%
86 }

```

```

87 \NewDocumentCommand \counterwithin {s0{\arabic}mm}{%
88 @ifbothcounters{#3}{#4}{%
89 @addtoreset{#3}{#4}%
90 \IfBooleanF #1{%
91 {\expandafter
92 \gdef\csname the#3\expandafter\endcsname
93 \expandafter
94 {\csname the#4\endcsname .#2{#3}}}}%
95 }%
96 }

97 </2ekernel | latexrelease>
98 <latexrelease>\EndIncludeInRelease
99 <latexrelease>\IncludeInRelease{2018-04-01}
100 <latexrelease> {\counterwithout}{counter without/within}%
101 <latexrelease>
102 <latexrelease>\def\counterwithout {\@ifstar\counterwithout@s\counterwithout@x}
103 <latexrelease>\def\counterwithout@s#1#2{%
104 <@ifbothcounters{#1}{#2}{\@removefromreset{#1}{#2}}}%
105 <latexrelease>\def\counterwithout@x#1#2{%
106 <@ifbothcounters{#1}{#2}{%
107 <latexrelease> {\@removefromreset{#1}{#2}}%
108 <latexrelease> \expandafter
109 <latexrelease> \gdef\csname the#1\expandafter\endcsname\expandafter
110 <latexrelease> {\expandafter
111 <latexrelease> \arabic\csname c@#1\endcsname}}%
112 <latexrelease>
113 <latexrelease>\def\counterwithin{\@ifstar\counterwithin@s\counterwithin@x}
114 <latexrelease>\def\counterwithin@s#1#2{%
115 <@ifbothcounters{#1}{#2}{\@addtoreset{#1}{#2}}}%

116 <latexrelease>\def\counterwithin@x#1#2{%
117 <@ifbothcounters{#1}{#2}{%
118 {\@addtoreset{#1}{#2}}%
119 \expandafter
120 \gdef\csname the#1\expandafter\endcsname\expandafter
121 <latexrelease> {\csname the#2\expandafter\endcsname\expandafter
122 <latexrelease> .\expandafter
123 <latexrelease> \arabic\csname c@#1\endcsname}}%
124 <latexrelease>
125 <latexrelease>\EndIncludeInRelease
126 <latexrelease>\IncludeInRelease{0000-00-00}
127 <latexrelease> {\counterwithout}{counter without/within}%
128 <latexrelease>\let \counterwithout \undefined
129 <latexrelease>\let \counterwithout@s \undefined
130 <latexrelease>\let \counterwithout@x \undefined
131 <latexrelease>\let \counterwithin \undefined
132 <latexrelease>\let \counterwithin@s \undefined
133 <latexrelease>\let \counterwithin@x \undefined
134 <latexrelease>\EndIncludeInRelease
135 <*2ekernel>

```

(End definition for \counterwithout and \counterwithin.)

Numbering commands for definitions of \theCOUNTER and \list arguments.  
All commands can now be used in text and math mode.

**\arabic** Representation of *⟨counter⟩* as arabic numerals. Changed 29 Apr 86 to make it print the obvious thing it COUNTER not positive.

136 \def\arabic#1{\expandafter\@arabic\csname c@#1\endcsname}

(*End definition for \arabic.*)

**\roman** Representation of *⟨counter⟩* as lower-case Roman numerals.

137 \def\roman#1{\expandafter\@roman\csname c@#1\endcsname}

(*End definition for \roman.*)

**\Roman** Representation of *⟨counter⟩* as upper-case Roman numerals.

138 \def\Roman#1{\expandafter\@Roman\csname c@#1\endcsname}

(*End definition for \Roman.*)

**\alph** Representation of *⟨counter⟩* as a lower-case letter: 1 = a, 2 = b, etc.

139 \def\alph#1{\expandafter\@alph\csname c@#1\endcsname}

(*End definition for \alph.*)

**\Alph** Representation of *⟨counter⟩* as an upper-case letter: 1 = A, 2 = B, etc.

140 \def\Alph#1{\expandafter\@Alph\csname c@#1\endcsname}

(*End definition for \Alph.*)

**\fnsymbol** Representation of *⟨COUNTER⟩* as a footnote symbol: 1 = \*, 2 = †, etc.

141 \def\fnsymbol#1{\expandafter\@fnsymbol\csname c@#1\endcsname}

(*End definition for \fnsymbol.*)

**\@arabic** \@arabic\FOOcounter Representation of \FOOcounter as arabic numerals.

142 \def\@arabic#1{\number #1} %% changed 29 Apr 86

(*End definition for \@arabic.*)

**\@roman** \@roman\FOOcounter Representation of \FOOcounter as lower-case Roman numerals.

143 \def\@roman#1{\romannumeral #1}

(*End definition for \@roman.*)

**\@Roman** \@Roman\FOOcounter Representation of \FOOcounter as upper-case Roman numerals.

144 \def\@Roman#1{\expandafter\@slowromancap\romannumeral #1@}

(*End definition for \@Roman.*)

**\@slowromancap** Fully expandable macro to change a roman number to uppercase.

145 \def\@slowromancap#1{\ifx @#1% then terminate

146 \else

147 \if i#1I\else\if v#1V\else\if x#1X\else\if l#1L\else\if

148 c#1C\else\if d#1D\else \if m#1M\else#1\fi\fi\fi\fi\fi\fi\fi\fi

149 \expandafter\@slowromancap

150 \fi

151 }

(*End definition for \@slowromancap.*)

\@alph \c@alph\FOOcounter Representation of \FOOcounter as a lower-case letter: 1 = a, 2 = b, etc.

```
152 \def\@alph#1{%
153 \ifcase#1\or a\or b\or c\or d\or e\or f\or g\or h\or i\or j\or
154 k\or l\or m\or n\or o\or p\or q\or r\or s\or t\or u\or v\or w\or x\or
155 y\or z\else\@ctrerr\fi}
```

(End definition for \@alph.)

\@Alph \c@Alph\FOOcounter Representation of \FOOcounter as an upper-case letter: 1 = A, 2 = B, etc.

```
156 \def\@Alph#1{%
157 \ifcase#1\or A\or B\or C\or D\or E\or F\or G\or H\or I\or J\or
158 K\or L\or M\or N\or O\or P\or Q\or R\or S\or T\or U\or V\or W\or X\or
159 Y\or Z\else\@ctrerr\fi}
```

(End definition for \@Alph.)

\@fnsymbol Typesetting old fashioned footnote symbols. This can be done both in text or math mode now.

This macro is another example of an ever recurring problem in TeX: Determining if something is text-mode or math-mode. It is imperative for the decision between text and math to be delayed until the actual typesetting is done as the code in question may go through an \edef or \write where an \ifmmode test would be executed prematurely. Hence in the implementation below, \@fnsymbol is not robust in itself but the parts doing the actual typesetting are.

In the case of \@fnsymbol we make use of the robust command \TextOrMath which takes two arguments and typesets the first if in text-mode and the second if in math-mode. Note that in order for this command to make the correct decision, it must insert a \relax token if run under regular TeX, which ruins any kerning between the preceding characters and whatever awaits typesetting. If you use eTeX as engine for L<sup>A</sup>T<sub>E</sub>X (as recommended) this unfortunate side effect is not present.

```
160 </2ekernel>
161 <|latexrelease|\IncludeInRelease{2015/01/01}{\@fnsymbol}{Use \TextOrMath}%
162 <*2ekernel | latexrelease>
163 \def\@fnsymbol#1{%
164 \ifcase#1\or \TextOrMath{textasteriskcentered}*\or
165 \TextOrMath{textdagger}\dagger\or
166 \TextOrMath{textdaggerdbl}\ddagger\or
167 \TextOrMath{textsection}\mathsection\or
168 \TextOrMath{textparagraph}\mathparagraph\or
169 \TextOrMath{textbardbl}\textbar\or
170 \TextOrMath{\textasteriskcentered}\textasteriskcentered\{**}\or
171 \TextOrMath{\textdagger}\textdagger\{dagger\dagger}\or
172 \TextOrMath{\textdaggerdbl}\textdaggerdbl\{ddagger\dagger}\else
173 \@ctrerr\fi
174 }%
175 </2ekernel | latexrelease>
176 <|latexrelease|\EndIncludeInRelease
177 <|latexrelease|\IncludeInRelease{0000/00/00}{\@fnsymbol}{Use \TextOrMath}%
178 <|latexrelease|\def\@fnsymbol#1{\ensuremath{%
179 <|latexrelease|\ \ifcase#1\or *\or \dagger\or \ddagger\or \mathsection\or
180 <|latexrelease|\mathparagraph\or \textbar\or **\or \dagger\dagger}
```

```

181 〈\latexrelease〉 \or \ddagger\ddagger \else\ctrerr\fi}}}%
182 〈\latexrelease〉\EndIncludeInRelease
183 〈*2ekernel〉

```

(End definition for \cfnsymbol.)

- \TextOrMath When using regular TeX, we make this command robust so that it always selects the correct branch in an \ifmmode switch with the usual disadvantage of ruining kerning. For the application we use it for here that shouldn't matter. The alternative would be to mimic \IeC from inputenc but then it will have the disadvantage of choosing the wrong branch if appearing at the beginning of an alignment cell. However, users of eTeX will be pleasantly surprised to get the best of both worlds and no bad side effects.

First some code for checking if we are running eTeX but making sure not to permanently turn \protected into \relax.

```

184 〈/2ekernel〉
185 〈\latexrelease〉\IncludeInRelease{2015/01/01}{\TextOrMath}{\TextOrMath}%
186 〈*2ekernel | latexrelease〉
187 \begingroup\expandafter\expandafter\expandafter\endgroup
188 \expandafter\ifx\csname protected\endcsname\relax

```

In case of ordinary TeX we define \TextOrMath as a robust command but make sure it always grabs its arguments. If we didn't do this it might very well gobble spaces in the input stream.

```

189 \DeclareRobustCommand\TextOrMath{%
190 \ifmmode \expandafter\@secondoftwo
191 \else \expandafter\@firstoftwo \fi}
192 \protected@edef\TextOrMath#1#2{\TextOrMath{#1}{#2}}
193 \else

```

For eTeX the situation is similar. The robust macro is a hidden one so that we again avoid problems of gobbling spaces in the input.

```

194 \protected\expandafter\def\csname TextOrMath\space\endcsname{%
195 \ifmmode \expandafter\@secondoftwo
196 \else \expandafter\@firstoftwo \fi}
197 \edef\TextOrMath#1#2{%
198 \expandafter\noexpand\csname TextOrMath\space\endcsname
199 {#1}{#2}}
200 \fi
201 〈/2ekernel | latexrelease〉
202 〈\latexrelease〉\EndIncludeInRelease
203 〈\latexrelease〉\IncludeInRelease{0000/00/00}{\TextOrMath}{\TextOrMath}%
204 〈\latexrelease〉\let\TextOrMath\@undefined
205 〈\latexrelease〉\EndIncludeInRelease
206 〈*2ekernel〉

```

(End definition for \TextOrMath.)

```
207 〈/2ekernel〉
```

# File u

## ltlength.dtx

### 1 Lengths

```
\newlength Declare #1 to be a new length command.
 \setlength Set the length command, #1, to the value #2.
 \addtolength Increase the value of the length command, #1, by the value #2.
 \settowidth Set the length, #1 to the width of a box containing #2.
 \settoheight Set the length, #1 to the height of a box containing #2.
 \settodepth Set the length, #1 to the depth of a box containing #2.
 1 <*2ekernel>
 2 \message{lengths,}

\newlength
 3 \def\newlength#1{\@ifdefinable#1{\newskip#1}}

(End definition for \newlength.)

\setlength
 4 </2ekernel>
 5 <latexrelease>\IncludeInRelease{2015/01/01}%
 6 <latexrelease> {\setlength}{Using \setlength with \dimen0}%
 7 <*2ekernel | latexrelease>
 8 \def\setlength#1#2{#1 #2\relax}
 9 </2ekernel | latexrelease>
10 <latexrelease>\EndIncludeInRelease
11 <latexrelease>\IncludeInRelease{0000/00/00}%
12 <latexrelease> {\setlength}{Using \setlength with \dimen0}%
13 <latexrelease>\def\setlength#1#2{#1#2\relax}
14 <latexrelease>\EndIncludeInRelease
15 <*2ekernel>

(End definition for \setlength.)

\addtolength \relax added 24 Mar 86
16 \def\addtolength#1#2{\advance#1 #2\relax}

(End definition for \addtolength.)

\settoheight The obvious analogs of \settowidth.
 \settodepth
 \settowidth
 @settodim Clear the memory afterwards (which might be a lot).
17 \def@settodim#1#2#3{\setbox@tempboxa\hbox{\#3}#2#1\@tempboxa
18 \setbox@tempboxa\box\voidb@x}
19 \DeclareRobustCommand\settoheight{\@settodim\ht}
20 \DeclareRobustCommand\settodepth {\@settodim\dp}
21 \DeclareRobustCommand\settowidth {\@settodim\wd}

(End definition for \settoheight and others.)
```

`\@settopoint` This macro takes the contents of the skip register that is supplied as its argument and removes the fractional part to make it a whole number of points. This can be used in class files to avoid values like 345.466666pt when calculating a dimension.

```
22 \def\@settopoint#1{\divide#1\p@\multiply#1\p@}
23 \end{macro}
```

(End definition for `\@settopoint`.)

# File v

## ltfssbas.dtx

This file contains the main implementation of the ‘low level’ font selection commands. See other parts of the L<sup>A</sup>T<sub>E</sub>X distribution, or *The L<sup>A</sup>T<sub>E</sub>X Companion* for higher level documentation of the L<sup>A</sup>T<sub>E</sub>X ‘New’ Font Selection Scheme.

**Warning:** The macro documentation is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

### 1 Preliminary macros

We define a number of macros that will be used later.

`\@nomath` `\@nomath` is used by most macros that will have no effect in math mode. It issues a warning message.

```
1 {*2ekernel}
2 \def\@nomath#1{\relax\ifmmode
3 \@font@warning{Command \noexpand#1invalid in math mode}\fi}
```

(End definition for `\@nomath`.)

`\no@alphabet@error` The macro `\no@alphabet@error` is called whenever the user requests a math *alphabet* that is not available in the current *version*. In math mode an error message is produced otherwise the command keeps silent. The argument is the name of the control sequence that identifies the math *alphabet*. The `\relax` at the beginning is necessary to prevent T<sub>E</sub>X from scanning too far in certain situations.

```
4 \gdef\no@alphabet@error#1{\relax\ifmmode
5 \@latex@error{Math\space alphabet\space identifier\space
6 \noexpand#1is\space undefined\space in\space math\space
7 version\space ‘\math@version’}%
8 {Your\space requested\space math\space alphabet\space
9 is\space undefined\space in\space the\space current\space
10 math\space version.^^JCheck\space the\space spelling\space
11 or\space use\space the\space \noexpand\SetMathAlphabet\space
12 command.}%
13 \fi}
```

(End definition for `\no@alphabet@error`.)

`\new@mathgroup` `\mathgroup` We also give a new name to `\newfam` and `\fam` to avoid verbal confusion (see the introduction).<sup>23</sup>

```
14 \%def\new@mathgroup{\alloc@8\mathgroup\chardef\sixt@n}
15 \let\mathgroup\fam
16 \%let\newfam\new@mathgroup
17 \onlypreamble\new@mathgroup
```

(End definition for `\new@mathgroup` and `\mathgroup`.)

---

<sup>23</sup>For the same reason it seems advisable to `\let\fam` and `\newfam` equal to `\relax`, but this is commented out to retain compatibility to existing style files.

## 2 Macros for setting up the tables

```
\DeclareFontShape{
18 \def\DeclareFontShape{\begingroup
First we restore the catcodes of all characters used in the syntax.
19 \nfss@catcodes
We use \expandafter \endgroup to restore catcode in case something goes wrong with
the argument parsing (suggested by Tim Van Zandt)
20 \expandafter\endgroup
21 \DeclareFontShape@}
(End definition for \DeclareFontShape.)

\DeclareFontShape@
22 {/2ekernel}
23 {*2ekernel | latexrelease}
24 {latexrelease} \IncludeInRelease{2020/02/02}{%
25 {latexrelease} {\DeclareFontShape@}{Maybe drop one m}%
26 \def\DeclareFontShape@#1#2#3#4#5#6{
27 \expandafter\ifx\csname #1#2\endcsname\relax
28 \@latex@error{Font family '#1#2' unknown}\@eha
29 \else
If the series value is incorrectly specified with an extra "m", e.g., "mc" instead of just
"c", drop the surplus "m" but keep the "m" if it is by its own. In that case also issue a
warning that the declaration needs correction.
For this we compare the given value #3 with one where we may have dropped an "m".
If nothing has changes, fine. Otherwise there was a wrong value which is now corrected
in \reservedb so we use that and also issue a warning.
30 \edef\reserved@a{#3}%
31 \series@maybe@drop@one@m\reserved@a\reserved@b
32 \ifx\reserved@a\reserved@b\else
33 \@latex@note{Font shape #1/#2/#3/#4 has incorrect series
34 value '#3'.\MessageBreak It should not contain an 'm'!
35 Please correct it.\MessageBreak Found} %
36 \fi
37 \expandafter
38 \xdef\csname#1/#2/\reserved@b/#4\endcsname
39 {\expandafter\noexpand\csname #5\endcsname} %
40 %
Most of the time #6 is empty so using \let to \empty saves on space compared to using
\def. That's really one of the old space saving techniques and probably not necessary
these days.
41 \def\reserved@a{#6}%
42 \global
43 \expandafter\let\csname#5\expandafter\endcsname
44 \ifx\reserved@a\empty
45 \empty
46 \else
47 \reserved@a
48 \fi
49 \fi
50 }
```

```

51 </2ekernel | latexrelease>
52 <latexrelease>\EndIncludeInRelease
53 <latexrelease>\IncludeInRelease{0000/00/00}%
54 <latexrelease> {\DeclareFontShape@}{Maybe drop one m}%
55 <latexrelease>
56 <latexrelease>\def\DeclareFontShape@#1#2#3#4#5#6{%
57 <latexrelease> \expandafter\ifx\csname #1+#2\endcsname\relax
58 <latexrelease> \@latex@error{Font family '#1+#2' unknown}\@eha
59 <latexrelease> \else
60 <latexrelease> \expandafter
61 <latexrelease> \xdef\csname#1/#2/#3/#4\endcsname{\expandafter\noexpand
62 <latexrelease> \csname #5\endcsname}%
63 <latexrelease> \def\reserved@a{#6}%
64 <latexrelease> \global
65 <latexrelease> \expandafter\let\csname#5\expandafter\endcsname
66 <latexrelease> \ifx\reserved@a\empty
67 <latexrelease> \empty
68 <latexrelease> \else
69 <latexrelease> \reserved@a
70 <latexrelease> \fi
71 <latexrelease> \fi
72 <latexrelease> }
73 <latexrelease>\EndIncludeInRelease
74 <*2ekernel>

```

(*End definition for \DeclareFontShape@.*)

\DeclareFixedFont Define a direct font switch that avoids all overhead.

```

75 \def\DeclareFixedFont#1#2#3#4#5#6{%
76 \begingroup
77 \math@fontsfalse
78 \every@math@size{}%
79 \fontsize{#6}\z@
80 \usefont{#2}{#3}{#4}{#5}%
81 \global\expandafter\let\expandafter#1\the\font
82 \endgroup
83 }

```

(*End definition for \DeclareFixedFont.*)

\do@subst@correction

```

84 \def\do@subst@correction{%
85 \xdef\subst@correction{%
86 \font@name
87 \global\expandafter\font
88 \csname \curr@fontshape/\f@size\endcsname
89 \noexpand\fontname\font
90 \relax}%

```

Calling \subst@correction after the current group means calling it after we have loaded the substitution font which is done inside a group.

```

91 \aftergroup\subst@correction
92 }

```

(*End definition for \do@subst@correction.*)

```
\DeclareFontFamily
```

```
93 \def\DeclareFontFamily#1#2#3{%
```

If we want fast checking for the encoding scheme we can just check for `\T@..` being defined.

```
94 % \@tempswafalse
95 % \def\reserved@b{#1}%
96 % \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
97 % \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
98 % \cdp@list
99 % \if@tempswa
100 \ifundefined{T@#1}%
101 {%
102 \@latex@error{Encoding scheme '#1' unknown}\@eha
103 }%
104 {%
```

Now we have to define the macro `\(#1)+(#2)` to contain #3. But since most of the time #3 will be empty we use `\let` in a tricky way rather than a simple `\def` since this will save internal memory. We store the argument #3 in a temporary macro `\reserved@a`.

```
105 \def\reserved@a{#3}%
```

We compare `\reserved@a` with `\@empty`. If these two are the same we `\let` the ‘extra’ macro equal to `\@empty` which is not the same as doing a `\let` to `\reserved@a` — the latter would blow one extra memory location rather than reusing the one from `\@empty`.

```
106 \global
107 \expandafter\let\csname #1+#2\expandafter\endcsname
108 \ifx \reserved@a\@empty
109 \@empty
110 \else \reserved@a
111 \fi
112 {%
113 }
```

(End definition for `\DeclareFontFamily`.)

`\cdp@list` We initialize the code page list to be empty.

```
114 \let\cdp@list\@empty
115 \onlypreamble\cdp@list
```

(End definition for `\cdp@list`.)

```
\cdp@elt
```

```
116 \let\cdp@elt\relax
117 \onlypreamble\cdp@elt
```

(End definition for `\cdp@elt`.)

```
\DeclareFontEncoding
```

```
118 \def\DeclareFontEncoding{%
```

First we start with ignoring all blanks and newlines since every surplus space in the second or third argument will come out in a weird place in the document.

```

119 \begingroup
120 \nfss@catcodes
121 \expandafter\endgroup
122 \DeclareFontEncoding@}
123 \onlypreamble\DeclareFontEncoding

124 \def\DeclareFontEncoding@#1#2#3{%
125 \expandafter
126 \ifx\csname T@#1\endcsname\relax
127 \def\cdp@elt{\noexpand\cdp@elt}%
128 \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
129 {\default@family}{\default@series}%
130 {\default@shape}}%

```

To support encoding dependent commands (like accents) we initialise the command  $\langle encoding \rangle$ -cmd to be  $\@changed@cmd$ . (See `ltoutenc.dtx` for details.)

```

131 \expandafter\let\csname#1-cmd\endcsname\@changed@cmd
132 \else
133 \@font@info{Redeclaring font encoding #1}%
134 \fi
135 \global\@namedef{T@#1}{#2}%
136 \global\@namedef{M@#1}{\default@M#3}%

```

Keep a record of the last encoding being declared:

```

137 \xdef\LastDeclaredEncoding{#1}%
138 }
139 \onlypreamble\DeclareFontEncoding@

```

(End definition for `\DeclareFontEncoding`.)

`\LastDeclaredEncoding` The last encoding being declared by `\DeclareFontEncoding`.

```
140 \def\LastDeclaredEncoding{}%
```

(End definition for `\LastDeclaredEncoding`.)

#### `\DeclareFontSubstitution`

```

141 \def\DeclareFontSubstitution#1#2#3#4{%
142 \expandafter
143 \ifx\csname T@#1\endcsname\relax
144 \@latex@error{Encoding scheme '#1' unknown}\@eha
145 \else
146 \begingroup

```

We loop through the `\cdp@list` and rebuild it anew in `\toks@` thereby replacing the defaults for the encoding in question with the new defaults. It is important to store the encoding to test against expanded in `\reserved@a` since it might just be `\LastDeclaredEncoding` that is passed as #1.

```

147 \edef\reserved@a{#1}%
148 \toks@{ }%
149 \def\cdp@elt##1##2##3##4{%
150 \def\reserved@b{##1}%
151 \ifx\reserved@a\reserved@b

```

Here we use the new defaults but we use ##1 (i.e., the encoding name already stored previously) since we know that it is expanded.

```
152 \addto@hook\toks@{\cdp@elt{##1}{##2}{##3}{##4}}%
153 \else
```

If \reserved@a and \reserved@b differ then we simply copy from the old list to the new.

```
154 \addto@hook\toks@{\cdp@elt{##1}{##2}{##3}{##4}}%
155 \fi}%
156 \cdp@list
157 \xdef\cdp@list{\the\toks@}%
158 \endgroup
159 \global
160 \cnamedef{D@#1}{%
161 \def\default@family{#2}%
162 \def\default@series{#3}%
163 \def\default@shape{#4}%
164 }%
165 \fi
166 }
167 \onlypreamble\DeclareFontSubstitution
```

(End definition for \DeclareFontSubstitution.)

### \DeclareFontEncodingDefaults

```
168 \def\DeclareFontEncodingDefaults#1#2{%
169 \ifx\relax#1\else
170 \ifx\default@T\empty\else
171 \font@info{Overwriting encoding scheme text defaults}%
172 \fi
173 \gdef\default@T{#1}%
174 \fi
175 \ifx\relax#2\else
176 \ifx\default@M\empty\else
177 \font@info{Overwriting encoding scheme math defaults}%
178 \fi
179 \gdef\default@M{#2}%
180 \fi
181 }
182 \onlypreamble\DeclareFontEncodingDefaults
```

(End definition for \DeclareFontEncodingDefaults.)

### \default@T

### \default@M

```
183 \let\default@T\empty
184 \let\default@M\empty
```

(End definition for \default@T and \default@M.)

### \DeclarePreloadSizes

```
185 \def\DeclarePreloadSizes#1#2#3#4#5{%
186 \@ifundefined{T@#1}%
187 {\@latex@error{Encoding scheme '#1' unknown}\@eha}%
188 {}%
```

Don't know at the moment what this group here does!

```
189 \begingroup
```

We define a macro `\reserved@f`<sup>24</sup> that grabs the next *size* and loads the corresponding font. This is done by delimiting `\reserved@f`'s only argument by the token `,` (comma).

```
190 \def\reserved@f##1,{%
```

The end of the list will be detected when there are no more elements, i.e. when `\reserved@f`'s argument is empty. The trick used here is explained in Appendix D of the TeXbook: if the argument is empty the `\if` will select the first clause and `\let` `\reserved@f` equal to `\relax`. (We use the `>` character here since it cannot appear in font file names.)

```
191 \if>##1>%
192 \let\reserved@f\relax
193 \else
```

Otherwise, we define `\font@name` appropriately and call `\pickup@font` to do the work. Note that the requested `\curr@fontshape` combination must have been defined, or you will get an error. The definition of `\font@name` is carried out globally to be consistent with the rest of the code in this file.

```
194 \xdef\font@name{\csname#1/#2/#3/#4/#1\endcsname}%
195 \pickup@font
```

Now we forget the name of the font just loaded. More precisely, we set the corresponding control sequence to `\relax`. This means that later on, when the font is first used, the macro `\define@newfont` is called again to execute the 'extra' macro for this font.

```
196 \global\expandafter\let\font@name\relax
197 \fi
```

Finally we call `\reserved@f` again to process the next *size*. If `\reserved@f` was `\let` equal to `\relax` this will end the macro.

```
198 \reserved@f}%
```

We finish with reinserting the list of sizes after the `\reserved@f` macro and appending an empty element so that the end of the list is recognized properly.

```
199 \reserved@f#5,%
200 \endgroup
201 }%
202 }
203 @onlypreamble\DeclarePreloadSizes
```

(End definition for `\DeclarePreloadSizes`.)

`\ifmath@fonts` We need a switch to decide if we have to switch math fonts. For this purpose we provide `\ifmath@fonts` that can be set to true or false by the `\S@...` macros depending on if math fonts are provided for this size or not. The default is of course to switch all fonts.

```
204 \newif\ifmath@fonts \math@fontstrue
```

(End definition for `\ifmath@fonts`.)

---

<sup>24</sup>We cannot use `\@tempa` since it is needed in `\pickup@font`.

\DeclareMathSizes \DeclareMathSizes takes the text size, math text size, math script size, and math scriptscript size as arguments and defines the right \S@... macro.

```

205 \def\DeclareMathSizes{%
206 \@ifstar{\@DeclareMathSizes\math@fontsfalse}{%
207 {\@DeclareMathSizes{}}}
208 \onlypreamble\DeclareMathSizes

```

(End definition for \DeclareMathSizes and \DeclareMathSizes\*.)

\@DeclareMathSizes This modification by Michael J. Downes on comp.text.tex on 2002/10/17 allows the user to have settings such as

```

\DeclareMathSizes{9.5dd}{9.5dd}{7.4dd}{6.6dd}.

```

```

209 </2ekernel>
210 <latexrelease>\IncludeInRelease[2015/01/01]{\@DeclareMathSizes}%
211 <latexrelease> {Arbitrary units in \DeclareMathSizes}%
212 <*2ekernel | latexrelease>
213 \def\@DeclareMathSizes #1#2#3#4#5{%
214 \@defaultunits\dimen@ #2pt\relax\@nnil
215 \if $#3$%
216 \expandafter\let\csname S@\strip@pt\dimen@\endcsname\math@fontsfalse
217 \else
218 \@defaultunits\dimen@ii #3pt\relax\@nnil
219 \@defaultunits\@tempdima #4pt\relax\@nnil
220 \@defaultunits\@tempdimb #5pt\relax\@nnil
221 \toks@{\#1}%
222 \expandafter\xdef\csname S@\strip@pt\dimen@\endcsname{%
223 \gdef\noexpand\tf@size{\strip@pt\dimen@ii}%
224 \gdef\noexpand\sf@size{\strip@pt\@tempdima}%
225 \gdef\noexpand\ssf@size{\strip@pt\@tempdimb}%
226 \the\toks@
227 }%
228 \fi
229 }%
230 </2ekernel | latexrelease>
231 <latexrelease>\EndIncludeInRelease
232 <latexrelease>\IncludeInRelease[0000/00/00]{\@DeclareMathSizes}%
233 <latexrelease> {Arbitrary units in \DeclareMathSizes}%
234 <latexrelease>\def\@DeclareMathSizes#1#2#3#4#5{%
235 \@defaultunits\dimen@#2pt\relax\@nnil
236 \if $#3$%
237 \expandafter \let
238 \csname S@\strip@pt\dimen@\endcsname
239 \math@fontsfalse
240 \else
241 \expandafter \gdef
242 \csname S@\strip@pt\dimen@\endcsname
243 {\gdef\tf@size{\#3}\gdef\sf@size{\#4}%
244 \gdef\ssf@size{\#5}%
245 #1%
246 \fi}%
247 \expandafter \gdef
248 \csname S@\strip@pt\dimen@\endcsname
249 {\gdef\tf@size{\#3}\gdef\sf@size{\#4}%

```

```
250 \onlypreamble\@DeclarMathSizes
```

(End definition for \@DeclarMathSizes.)

## 3 Selecting a new font

### 3.1 Macros for the user

```
\fontencoding
\f@encoding
```

As we said in the introduction a font is described by four parameters. We first define macros to specify the wanted *family*, *series*, or *shape*. These are simply recorded in internal macros `\f@family`, `\f@series`, and `\f@shape`, resp. We use `\edef`'s so that the arguments can also be macros.

```
251 \DeclareRobustCommand\fontencoding[1]{%
252 \expandafter\ifx\csname T@\#1\endcsname\relax
253 \@latex@error{Encoding scheme '#1' unknown}\@eha
254 \else
255 \edef\f@encoding{\#1}%
256 \ifx\cf@encoding\f@encoding
```

If the new encoding is the same as the old encoding we have nothing to do. However, in case we had a sequence of several encoding changes without a `\selectfont` in-between we can save processing by making sure that `\enc@update` is `\relax`.

```
257 \let\enc@update\relax
258 \else
```

If current and new encoding differ we define the macro `\enc@update` to contain all updates necessary at `\selectfont` time.

```
259 \let\enc@update\@enc@update
260 \fi
261 \fi
262 }
```

(End definition for `\fontencoding` and `\f@encoding`.)

```
\@enc@update
```

```
263 \def\@enc@update{%
```

When `\@enc@update` is executed `\f@encoding` holds the encoding name for the new encoding and `\cf@encoding` the name of the last active encoding.

We start by setting the init command for encoding dependent macros to `\@changed@cmd`.

```
264 \expandafter
265 \let
266 \csname\cf@encoding -cmd\endcsname
267 \@changed@cmd
```

Then we turn the one for the new encoding to `\@current@cmd` (see `ltoutenc.dtx` for further explanations).

```
268 \expandafter
269 \let
270 \csname\f@encoding-cmd\endcsname
271 \@current@cmd
```

We execute the default settings `\default@T`, followed by the one for the new encoding.

```
272 \default@T
273 \csname T@\f@encoding\endcsname
```

Finally we change the default substitution values, disable `\enc@update` and make `\f@encoding` officially the current encoding.

```
274 \csname D@\f@encoding\endcsname
275 \let\enc@update\relax
276 \let\cf@encoding\f@encoding
277 }
```

*(End definition for \@@enc@update.)*

`\enc@update` The default action in `\selectfont` is to do nothing.

```
278 \let\enc@update\relax
```

*(End definition for \enc@update.)*

`\fontfamily`

`\f@family`

`\fontseries`

`\f@series`

`\fontshape`

`\f@shape`

```
279 \DeclareRobustCommand\fontfamily[1]{\edef\f@family{#1}}
```

There are now defined later (and differently).

```
280 \% \DeclareRobustCommand\fontseries[1]{\edef\f@series{#1}}
```

```
281 \% \DeclareRobustCommand\fontshape [1]{\edef\f@shape{#1}}
```

*(End definition for \fontfamily and others.)*

`\usefont` Some handy abbreviation if you want to get some particular font in the current size. If also the size should change one has to issue a `\fontsize` command first.

`\fontencoding` needs to do some setup work so we call that, but instead of calling `\fontfamily`, `\fontseries` and `\fontshape` it earlier versions of this code did, we now set `\f@family`, etc. directly. If we would call `\fontseries` or `\fontshape` as it was done in the past, they would now interact with the existing series and shape which is not desired if we intend to use an explicit font shape!

```
282 </2ekernel>
283 <*2ekernel | latexrelease>
284 <latexrelease>\IncludeInRelease{2021/06/01}%
285 <latexrelease> {\usefont}{Force font face}%
286 \DeclareRobustCommand\usefont[4]{\fontencoding{#1}%
287 \edef\f@family{#2}%
288 \set@target@series{#3}%
289 \edef\f@shape{#4}%

```

Any earlier `\fontseries`, etc. should be canceled and we should switch unconditionally to the requested font face so we drop any code that may have been stored in `\delayed@f@adjustment`.

```
290 \let\delayed@f@adjustment\empty
291 \selectfont
292 \ignorespaces}
293 </2ekernel | latexrelease>
294 <latexrelease>\EndIncludeInRelease
295 <latexrelease>\IncludeInRelease{2020/02/02}%
296 <latexrelease> {\usefont}{Drop m in usefont}%
297 <latexrelease>
298 <latexrelease>\DeclareRobustCommand\usefont[4]{\fontencoding{#1}%
299 <latexrelease> \edef\f@family{#2}%
300 <latexrelease> \set@target@series{#3}%
301 <latexrelease> \edef\f@shape{#4}\selectfont
```

```

302 <|latexrelease> \ignorespaces}
303 <|latexrelease>
304 <|latexrelease>\EndIncludeInRelease
305 <|latexrelease>\IncludeInRelease{0000/00/00}%
306 <|latexrelease> {\usefont}{Drop m in usefont}%
307 <|latexrelease>
308 <|latexrelease>\DeclareRobustCommand\usefont[4]{\fontencoding{\#1}%
309 <|latexrelease> \edef\f@family{\#2}%
310 <|latexrelease> \edef\f@series{\#3}%
311 <|latexrelease> \edef\f@shape{\#4}\selectfont
312 <|latexrelease> \ignorespaces}
313 <|latexrelease>
314 <|latexrelease>\EndIncludeInRelease
315 {*2ekernel}

```

(*End definition for \usefont.*)

**\linespread** The command `\linespread` changes the current `\baselinestretch` by calling `\set@fontsize`. The values for `\f@size` and `\f@baselineskip` will be left unchanged.

```

316 \DeclareRobustCommand\linespread[1]
317 {\set@fontsize{\#1}\f@size\f@baselineskip}

```

(*End definition for \linespread.*)

**\fontsize** We also define a macro that allows to specify a size. In this case, however, we also need the value of `\baselineskip`. As the first argument to `\set@fontsize` we pass the current value of `\baselinestretch`. This will either match the internal value (in which case nothing changes, or it will be an updated value due to a user change of that macro using `\renewcommand`. If we would pass the internal `\f@linespread` such a change would be effectively overwritten by a size change.

```

318 \DeclareRobustCommand\fontsize[2]
319 {\set@fontsize\baselinestretch{\#1}{\#2}}

```

(*End definition for \fontsize.*)

**\f@linespread** This macro holds the current internal value for `\baselinestretch`.

```

320 \let\f@family\empty
321 \let\f@series\empty
322 \let\f@shape\empty
323 \let\f@size\empty
324 \let\f@baselineskip\empty
325 \let\f@linespread\empty

```

(*End definition for \f@linespread.*)

**\cf@encoding**

```

326 \let\f@encoding\empty
327 \let\cf@encoding\empty

```

(*End definition for \cf@encoding.*)

\@defaultunits The function \@defaultunits when wrapped around a dimen or skip assignment supplies default units. Usage:

```
\@defaultunits\dimen@=#1pt\relax\@nnil
```

Note: the \relax is \*important\*. Other units can be substituted for the ‘pt’ if desired.

We use \remove@to@nnil as an auxiliary macros for \@defaultunits. It just has to gobble the supplied default unit ‘pt’ or whatever, if it wasn’t used in the assignment.

```
328 \def\@defaultunits{\afterassignment\remove@to@nnil}
```

(End definition for \@defaultunits.)

\strip@pt This macro strips the characters pt produced by using \the on a dimen register.

```
\rem@pt 329 \begingroup
330 \catcode`P=12
331 \catcode`T=12
332 \lowercase{
333 \def\x{\def\rem@pt##1.##2PT{##1\ifnum##2>\z@.##2\fi}}}
334 \expandafter\endgroup\x
335 \def\strip@pt{\expandafter\rem@pt\the}
```

(End definition for \strip@pt and \rem@pt.)

\mathversion \mathversion takes the math *version* name as argument, defines \math@version appropriately and switches to the font selected forcing a call to \glb@settings if the *version* is known to the system.

```
336 \DeclareRobustCommand\mathversion[1]
337 {\@nomath\mathversion
338 \expandafter\ifx\csname mv@\#1\endcsname\relax
339 \@latex@error{Math version '#1' is not defined}\@eha\else
340 \edef\math@version{\#1}\%
```

We need to force a math font setup both now and at the point where we return to the previous math version. Forcing a math font setup can simply be done by setting \glb@currsize to an invalid value since this will trigger the setup when the formula starts.

```
341 \gdef\glb@currsize{}%
```

When the scope of the current \mathversion ends we need to restore the old setup. However this time we need to force it directly at least if we are inside math, otherwise we could wait. Another way to enhance this code here is todo the setting only if the version really has changed after all. This might be interesting in case of amstext and boldsymbol.

```
342 \aftergroup\glb@settings
343 \fi}
```

(End definition for \mathversion and \math@version.)

If TeX would support a hook just before the end of a formula (opposite of \everymath so to speak) the implementation of the algorithm would be much simpler because in that case we would set up the correct math fonts at this point without having to worry about incorrect settings due to nesting. The same would be true if in L<sup>A</sup>T<sub>E</sub>X the use of \$ (as the primitive TeX command) would be impossible and instead only a higher-level interface would be available. Note that this does not mean that a \$ couldn’t be the short-hand

for starting and stopping that higher-level interface, it only means that the direct T<sub>E</sub>X function must be hidden.

Anyway, since we don't have this and won't have it in L<sub>A</sub>T<sub>E</sub>X 2<sub>E</sub> we need to implement it in a somewhat slower way.

We test for the current math font setup on entry of a formula, i.e., on the hooks `\everymath` and `\everydisplay`. But since these hooks may contain user data we provide ourselves with an internal version of these hooks which stays frozen.

|                                                                         |                                                                                                            |
|-------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| <code>\frozen@everymath</code>                                          | New internal names for <code>\everymath</code> and <code>\everydisplay</code> .                            |
| <code>\frozen@everydisplay</code>                                       | <code>344 \let\frozen@everymath\everymath</code><br><code>345 \let\frozen@everydisplay\everydisplay</code> |
| <i>(End definition for \frozen@everymath and \frozen@everydisplay.)</i> |                                                                                                            |
| <code>\everymath</code>                                                 | Now we provide now user hooks that will be called in the frozen internals.                                 |
| <code>\everydisplay</code>                                              | <code>346 \newtoks\everymath</code><br><code>347 \newtoks\everydisplay</code>                              |
| <i>(End definition for \everymath and \everydisplay.)</i>               |                                                                                                            |
| <code>\frozen@everymath</code>                                          | Now we define the behaviour of the frozen hooks: first check the math setup then call the user hook.       |
|                                                                         | <code>348 \frozen@everymath = {\check@mathfonts</code><br><code>349 \the\everymath}</code>                 |
| <i>(End definition for \frozen@everymath.)</i>                          |                                                                                                            |
| <code>\frozen@everydisplay</code>                                       | Ditto for the display hook.                                                                                |
|                                                                         | <code>350 \frozen@everydisplay = {\check@mathfonts</code><br><code>351 \the\everydisplay}</code>           |
| <i>(End definition for \frozen@everydisplay.)</i>                       |                                                                                                            |
| <code>\curr@math@size</code>                                            | This holds locally the current math size.                                                                  |
|                                                                         | <code>352 \let\curr@math@size\empty</code>                                                                 |
| <i>(End definition for \curr@math@size.)</i>                            |                                                                                                            |

### 3.2 Macros for loading fonts

`\pickup@font` The macro `\pickup@font` which is used in `\selectfont` is very simple: if the font name is undefined (i.e. not known yet) it calls `\define@newfont` to load it.

```
353 \def\pickup@font{%
354 \expandafter \ifx \font@name \relax
355 \define@newfont
356 \fi}
```

*(End definition for \pickup@font.)*

\split@name \pickup@font assumes that \font@name is set but it is sometimes called when \f@family, \f@series, \f@shape, or \f@size may have the wrong settings (see, e.g., the definition of \getanddefine@fonts). Therefore we need a macro to extract font *family*, *series*, *shape*, and *size* from the font name. To this end we define \split@name which takes the font name as a list of characters of \catcode 12 (without the backslash at the beginning) delimited by the special control sequence \nil. This is not very complicated: we first ensure that / has the right \catcode

357 {\catcode`/=12

and define \split@name so that it will define our private \f@encoding, \f@family, \f@series, \f@shape, and \f@size macros.

358 \gdef\split@name{\#1/#2/#3/#4/#5@\nil{\def\f@encoding{\#1}%
359 \def\f@family{\#2}%
360 \def\f@series{\#3}%
361 \def\f@shape{\#4}%
362 \def\f@size{\#5}}}

(End definition for \split@name.)

\curr@fontshape Abbreviation which may get removed again for speed.

363 \def\curr@fontshape{\f@encoding/\f@family/\f@series/\f@shape}

(End definition for \curr@fontshape.)

\define@newfont Now we can tackle the problem of defining a new font.

364 \def\define@newfont{%

We have already mentioned that the token list that \split@name will get as argument must not start with a backslash. To reach this goal we will set the \escapechar to -1 so that the \string primitive will not generate an escape character. To keep this change local we open a group. We use \begingroup for this purpose since \define@newfont might be called in math mode, and an empty \bgroup...\egroup would add an empty Ord atom to the math list and thus affect the spacing.

Also locally redefine \typeout so that ‘No file ...fd’ Warnings become Font Info message just sent to the log file.

365 \begingroup
366 \let\typeout\@font@info
367 \escapechar`m@ne

Then we extract *encoding scheme*, *family*, *series*, *shape*, and *size* from the font name. Note the four \expandafter's so that \font@name is expanded first, then \string, and finally \split@name.

368 \expandafter\expandafter\expandafter
369 \split@name\expandafter\string\font@name\@nil

If the \curr@fontshape combination is not available, (i.e. undefined) we call the macro \wrong@fontshape to take care of this case. Otherwise \extract@font will load the external font for us.

370 % \expandafter\ifx
371 % \csname\curr@fontshape\endcsname \relax
372 \try@load@fontshape % try always
373 % \fi
374 \expandafter\ifx
375 \csname\curr@fontshape\endcsname \relax
376 \wrong@fontshape\else

To allow substitution we call the `curr@fontshape` macro which usually will expand to `\relax` but may hold code for substitution (see `\subst@fontshape` definition).

```
377 % \csname\curr@fontshape\endcsname
378 \extract@font\fi
```

We are nearly finished and must only restore the `\escapechar` by closing the group.

```
379 \endgroup}
380 \def\try@load@fontshape{%
381 \expandafter
382 \ifx\csname\f@encoding+\f@family\endcsname\relax
383 \@font@info{Trying to load font information for
384 \f@encoding+\f@family}%
385 }
```

We predefine this combination to be `\empty` which means that next time we don't try again unnecessary in case we don't find a `.fd` file. If the file contains a `\DeclareFontFamily` command than this setting will be overwritten.

```
385 \global\expandafter\let
386 \csname\f@encoding+\f@family\endcsname\empty
```

Set the catcodes used in the syntax, but do it only once (this will be restored at the end of the font loading group).

```
387 \nfss@catcodes
388 \let\nfss@catcodes\relax
```

For increased portability make the external filename monocase, but look for the (old style) mixed case filename if the first attempt fails.

On any monocase system this means that the file is looked for twice which takes up time and string space, but at least for this release Check for both names to give people time to re-install their private fd files with lowercase names.

```
389 \edef\reserved@a{%
390 \lowercase{%
391 \noexpand\InputIfFileExists{\f@encoding\f@family.fd}}{%
392 \reserved@a\relax
393 {\@input{\f@encoding\f@family.fd}}{%
394 \fi}}
```

*(End definition for `\define@newfont`.)*

`\nfss@catcodes` This macro should contain the standard `\catcode` assignments to all characters which are used in the commands found in an `.fd` file and which might have special `\catcodes` in the middle of a document. If necessary, this list can be extended in a package file using a suitable number of `\expandafter`, i.e.,

```
\expandafter\def\expandafter\nfss@catcodes
\expandafter{\nfss@catcodes <additional settings>}
```

Note, that this macro might get executed several times since it is also called by `\DeclareFontShape`, thus it probably should not be misused as a general purpose hook.

```
395 \def\nfss@catcodes{%
```

We start by making @ a letter and ignoring all blanks and newlines.

```
396 \makeatletter
397 \catcode`\ 9%
398 \catcode`\^\I9%
399 \catcode`\^\M9%
```

Then we set up \, {, }, # and % in case an .fd file is loaded during a verbatim environment.

```
400 \catcode`\\z@
401 \catcode`{\@ne
402 \catcode`}`tw@
403 \catcode`\#6%
404 \catcode`\^7%
405 \catcode`\%14%
```

The we make sure that the important syntax parts have the right \catcode.

```
406 \@makeother\<%
407 \@makeother\>%
408 \@makeother*%
409 \@makeother\.%
410 \@makeother\-%
411 \@makeother\/%
412 \@makeother\[%
413 \@makeother\]%
```

(End definition for \nfss@catcodes.)

\LoadFontDefinitionFile Load and .fd files for some encoding and family (if it exists).

```
418 </2ekernel>
419 <*2ekernel | latexrelease>
420 <latexrelease>\IncludeInRelease{2020/02/02}%
421 <latexrelease> {\LoadFontDefinitionFile}{Loading .fd files}%
422 \def\LoadFontDefinitionFile#1#2{
423 \begingroup
424 \edef\f@encoding{#1}%
425 \edef\f@family{#2}%
426 \try@load@fontshape
427 \endgroup
428 }
429 </2ekernel | latexrelease>
430 <latexrelease>\EndIncludeInRelease
431 <latexrelease>\IncludeInRelease{0000/00/00}%
432 <latexrelease> {\LoadFontDefinitionFile}{Loading .fd files}%
433 <latexrelease>
434 <latexrelease>\let\LoadFontDefinitionFile\@undefined
435 <latexrelease>\EndIncludeInRelease
436 <*2ekernel>
```

(End definition for \LoadFontDefinitionFile.)

\DeclareFontFamilySubstitution The idea for this macro is stolen from the `substitutefont` package by Günter Milde, with some modifications and a new name.

Its purpose is to provide characters in a special encoding that are not available in the current font family to be taken from a different family that is visually compatible (or not if you choose badly). For example, you can match the GFS Didot Greek characters with TeX Gyre Pagella (Palatino) by specifying

```
\DeclareFontFamilySubstitution{LGR}{qpl}{udidot}
```

This way if you ask for the LGR encoding in for the qpl family you get the characters from the udot family substituted.

We need to ensure that the macro is defined with `\nfss@catcodes` in force (not quite sure why at the moment to be honest).

```
437 </2ekernel>
438 <*2ekernel | latexrelease>
439 <latexrelease>\IncludeInRelease{2020/02/02}%
440 <latexrelease> {\DeclareFontFamilySubstitution}{Provide family substitution}%
441 \begingroup
442 \nfss@catcodes
443 \gdef\DeclareFontFamilySubstitution#1#2#3{%
```

We only provide a set of silent substitutions. The package also (re)declared the family, but this is incorrect in my eyes and it is better to handle that differently.

Of course the families may still need loading at this point and so we arrange for this. Otherwise we might run into trouble because the necessary `\DeclareFontFamily` has not been seen.

```
444 \LoadFontDefinitionFile{#1}{#2}%
445 \LoadFontDefinitionFile{#1}{#3}%
446 \DeclareFontShape{#1}{#2}{m}{it}{<->ssub * #3/m/it}{}%
447 \DeclareFontShape{#1}{#2}{m}{n}{<->ssub * #3/m/n}{}%
448 \DeclareFontShape{#1}{#2}{m}{sc}{<->ssub * #3/m/sc}{}%
449 \DeclareFontShape{#1}{#2}{m}{sl}{<->ssub * #3/m/sl}{}%
```

These days a few more shapes might be around, so we declare those too. If they don't exist then after the first substitution normal fallbacks will happen.

```
450 \DeclareFontShape{#1}{#2}{m}{sw}{<->ssub * #3/m/sw}{}%
451 \DeclareFontShape{#1}{#2}{m}{scit}{<->ssub * #3/m/scit}{}%
452 \DeclareFontShape{#1}{#2}{m}{scsl}{<->ssub * #3/m/scsl}{}%
```

Same game with b and bx, for other weights you are on your own:

```
453 \DeclareFontShape{#1}{#2}{b}{it}{<->ssub * #3/b/it}{}%
454 \DeclareFontShape{#1}{#2}{b}{n}{<->ssub * #3/b/n}{}%
455 \DeclareFontShape{#1}{#2}{b}{scit}{<->ssub * #3/b/scit}{}%
456 \DeclareFontShape{#1}{#2}{b}{scsl}{<->ssub * #3/b/scsl}{}%
457 \DeclareFontShape{#1}{#2}{b}{sc}{<->ssub * #3/b/sc}{}%
458 \DeclareFontShape{#1}{#2}{b}{sl}{<->ssub * #3/b/sl}{}%
459 \DeclareFontShape{#1}{#2}{b}{sw}{<->ssub * #3/b/sw}{}%
460 \DeclareFontShape{#1}{#2}{bx}{it}{<->ssub * #3/bx/it}{}%
461 \DeclareFontShape{#1}{#2}{bx}{n}{<->ssub * #3/bx/n}{}%
462 \DeclareFontShape{#1}{#2}{bx}{scit}{<->ssub * #3/bx/scit}{}%
463 \DeclareFontShape{#1}{#2}{bx}{scsl}{<->ssub * #3/bx/scsl}{}%
464 \DeclareFontShape{#1}{#2}{bx}{sc}{<->ssub * #3/bx/sc}{}%
465 \DeclareFontShape{#1}{#2}{bx}{sl}{<->ssub * #3/bx/sl}{}%
466 \DeclareFontShape{#1}{#2}{bx}{sw}{<->ssub * #3/bx/sw}{}%
467 }
468 \endgroup
469 </2ekernel | latexrelease>
470 <latexrelease>\EndIncludeInRelease
471 <latexrelease>\IncludeInRelease{0000/00/00}%
472 <latexrelease> {\DeclareFontFamilySubstitution}{Provide family substitution}%
473 <latexrelease>
```

```
474 <latexrelease>\let\DeclareFontFamilySubstitution\@undefined
475 <latexrelease>\EndIncludeInRelease
476 <*2ekernel>
```

(End definition for \DeclareFontFamilySubstitution.)

- \DeclareErrorFont** Declare the last resort shape! We assume that in this fontshape there is a 10pt font but it doesn't really matter. We only loose one macro name if the assumption is false. But at least the font should be there!

```
477 </2ekernel>
478 <*2ekernel | latexrelease>
479 <latexrelease>\IncludeInRelease{2019/10/01}%
480 <latexrelease> {\DeclareErrorFont}{No side effects please}%
481 \def\DeclareErrorFont#1#2#3#4#5{%
482 \xdef\error@fontshape{%
483 \noexpand\expandafter\noexpand\split@name\noexpand\string
484 \expandafter\noexpand\csname#1/#2/#3/#4/#5\endcsname
485 \noexpand\@nil}%

```

Initialize all those internal variables which may or may not have values in the first seconds of NFSS' bootstrapping process. Later on such values will be updated when an encoding is selected, etc.

We definitely don't want to set `\f@encoding`; we can set all the others since if they are left "blank" any selection would grab "error default values" as well. However, this probably should go also—and now it did.

```
486 % \gdef\f@encoding{#1}%
487 \gdef\default@family{#2}%
488 \gdef\default@series{#3}%
489 \gdef\default@shape{#4}%
490 }
491 </2ekernel | latexrelease>
492 <latexrelease>\EndIncludeInRelease
493 <latexrelease>\IncludeInRelease{0000/00/00}%
494 <latexrelease> {\DeclareErrorFont}{No side effects please}%
495 <latexrelease>
496 <latexrelease>\def\DeclareErrorFont#1#2#3#4#5{%
497 <latexrelease> \xdef\error@fontshape{%
498 <latexrelease> \noexpand\expandafter\noexpand\split@name\noexpand\string
499 <latexrelease> \expandafter\noexpand\csname#1/#2/#3/#4/#5\endcsname
500 <latexrelease> \noexpand\@nil}%
501 <latexrelease> \gdef\default@family{#2}%
502 <latexrelease> \gdef\default@series{#3}%
503 <latexrelease> \gdef\default@shape{#4}%
504 <latexrelease> \global\let\f@family\default@family
505 <latexrelease> \global\let\f@series\default@series
506 <latexrelease> \global\let\f@shape\default@shape
507 <latexrelease> \gdef\f@size{#5}%
508 <latexrelease> \gdef\f@baselineskip{#5pt}%
509 <latexrelease> }
510 <latexrelease>\EndIncludeInRelease
511 <*2ekernel>
512 @onlypreamble\DeclareErrorFont
```

(End definition for \DeclareErrorFont.)

\wrong@fontshape Before we come to the macro \extract@font we have to take care of unknown \curr@fontshape combinations. The general strategy is to issue a warning and to try a default *shape*, then a default *series*, and finally a default *family*. If this last one also fails TeX will go into an infinite loop. But if the defaults are set incorrectly one deserves nothing else!

```

513 </2ekernel>
514 <|latexrelease>\IncludeInRelease{2015/01/01}{\wrong@fontshape}%
515 <|latexrelease> {Font substitution in preamble}%
516 <*2ekernel | latexrelease>
517 \def\wrong@fontshape{%
518 \csname D@f@encoding\endcsname % install defaults if in math

```

We remember the wanted \curr@fontshape combination which we will need in a moment.

```

519 \edef\reserved@a{\csname\curr@fontshape\endcsname}%
520 \ifx\last@fontshape\reserved@a
521 \errmessage{Corrupted NFSS tables}%
522 \error@fontshape
523 \else

```

Then we warn the user about the mess and set the shape to its default.

```
524 \let\f@shape\default@shape
```

If the combination is not known, try the default *series*.

```

525 \expandafter\ifx\csname\curr@fontshape\endcsname\relax
526 \let\f@series\default@series

```

If this is still undefined, try the default *family*. Otherwise give up. We never try to change the encoding scheme!

```

527 \expandafter
528 \ifx\csname\curr@fontshape\endcsname\relax
529 \let\f@family\default@family

```

If we change the font family and we are in the preamble then the corresponding .fd file may not been loaded yet. Therefore we try this now. Otherwise equating the requested font shape with the finally selected fontshape below will fail and can result in “NFSS tables corrupted”. After begin document that will not happen as all .fd files involved in substitution are loaded at \begin{document}.

```

530 \begingroup
531 \try@load@fontshape
532 \endgroup
533 \fi \fi
534 \fi

```

At this point a valid \curr@fontshape combination must have been found. We inform the user about this fact.

The \expandafter\string here stops TeX adding the space that it usually puts after command names in messages. The similar construction with \undefined just produces ‘undefined’, but saves a few tokens.

\@wrong@font@char is locally redefined in \UseTextSymbol from its normal (empty) definition, to report the symbol generating the font switch.

```

535 \@font@warning{Font shape `\\expandafter\\string\\reserved@a'
536 \\expandafter\\@gobble\\string\\undefined\\MessageBreak
537 using '\\curr@fontshape' instead\\@wrong@font@char}%
538 \\global\\let\\last@fontshape\\reserved@a

```

We change `\@defaultsubs` to produce a warning at the end of the document. The macro `\@defaultsubs` is initially `\relax` but gets changed here if some default font substitution happens. It is then executed in `\enddocument`.

```
539 \gdef\@defaultsubs{%
540 \@font@warning{Some font shapes were not available, defaults
541 substituted.\@gobbletwo}}%
```

If we substitute a `\curr@fontshape` combination by the default one we don't want the warning to be printed out whenever this (unknown) combination is used. Therefore we globally `\let` the macro corresponding to the wanted combination equal to its substitution. This requires the use of four `\expandafter`'s since `\csname... \endcsname` has to be expanded before `\reserved@a` (i.e. the requested combination), and this must happen before the `\let` is executed.

```
542 \global\expandafter\expandafter\expandafter\let
543 \expandafter\reserved@a
544 \csname\curr@fontshape\endcsname
```

Now we can redefine `\font@name` accordingly. This *must* be done globally since it might occur in the group opened by `\define@newfont`. If we would this definition were local the closing `\endgroup` there would restore the old meaning of `\font@name` and then switch to the wrong font at the end of `\selectfont` although the correct font was loaded.

```
545 \xdef\font@name{%
546 \csname\curr@fontshape\f@size\endcsname}%
```

The last thing this macro does is to call `\pickup@font` again to load the font if it is not defined yet. At this point this code will loop endlessly if the defaults are not well defined.

```
547 \pickup@font
548 {/2ekernel | latexrelease}
549 \EndIncludeInRelease
550 \IncludeInRelease{0000/00/00}{\wrong@fontshape}%
551 {Font substitution in preamble}%
552 \def\wrong@fontshape{%
553 \csname D@\f@encoding\endcsname
554 \edef\reserved@a{\csname\curr@fontshape\endcsname}%
555 \ifx\last@fontshape\reserved@a
556 \errmessage{Corrupted NFSS tables}%
557 \error@fontshape
558 \else
559 \let\f@shape\default@shape
560 \expandafter\ifx\csname\curr@fontshape\endcsname\relax
561 \Let\f@series\default@series
562 \expandafter
563 \ifx\csname\curr@fontshape\endcsname\relax
564 \let\f@family\default@family
565 \fi \fi
566 \fi
567 \font@warning{Font shape
568 '\expandafter\string\reserved@a'
569 \expandafter\@gobble\string\@undefined
570 \MessageBreak
571 using '\curr@fontshape' instead\@wrong@font@char}%
572 \global\let\last@fontshape\reserved@a
573 \gdef\@defaultsubs{%
574 \@font@warning{Some font shapes were not available,
```

```

575 ⟨latexrelease⟩ defaults substituted.\@gobbletwo} }%
576 ⟨latexrelease⟩ \global\expandafter\expandafter\expandafter\let
577 ⟨latexrelease⟩ \expandafter\reserved@a
578 ⟨latexrelease⟩ \csname\curr@fontshape\endcsname
579 ⟨latexrelease⟩ \xdef\font@name{%
580 ⟨latexrelease⟩ \csname\curr@fontshape/\f@size\endcsname}%
581 ⟨latexrelease⟩ \pickup@font}
582 ⟨latexrelease⟩\EndIncludeInRelease
583 {*2ekernel}

```

(End definition for `\wrong@fontshape`.)

`\@wrong@font@char` Normally empty but redefined in `\UseTextSymbol` so that the Font shape undefined message can refer to the symbol causing the problem.

```
584 \let\@wrong@font@char\@empty
```

(End definition for `\@wrong@font@char`.)

`\@@defaultsubs` See above.

```
585 \let\@defaultsubs\relax
```

(End definition for `\@@defaultsubs` and `\@defaultsubs`.)

`\strip@prefix` In `\extract@font` we will need a way to recover the replacement text of a macro. This is done by the primitive `\meaning` together with the macro `\strip@prefix` (for the details see appendix D of the TeXbook, p. 382).

```
586 \def\strip@prefix#1>{}
```

(End definition for `\strip@prefix`.)

## 4 Assigning math fonts to *versions*

`\install@mathalphabet` This is just another name for `\gdef` but we can redefine it if necessary later on.

```
587 \let\install@mathalphabet\gdef
```

(End definition for `\install@mathalphabet`.)

`\math@fonts`

```
588 \let\math@fonts\@empty
```

(End definition for `\math@fonts`.)

`\select@group` `\select@group` has four arguments: the new *⟨math alphabet identifier⟩* (a control sequence), the *⟨math group number⟩*, the extra macro for math mode and the `\curr@fontshape` definition macro name. We first check if we are in math mode.

```
589 \% \def\select@group#1#2#3{\relax\ifmmode
```

We do these things locally using `\begingroup` instead of `\bgroup` to avoid the appearance of an empty Ord atom on the math list.

```
590 \% \begingroup
```

We set the math fonts for the *family* in question by calling `\getanddefine@fonts` in the correct environment.

```
591 % \escapechar\m@ne
592 % \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
```

We globally select the math fonts...

```
593 % \globaldefs\one \math@fonts
... and close the group to restore \globaldefs and \escapechar.
594 % \endgroup
```

As long as no *size* or *version* change occurs the  $\langle\mathit{math alphabet identifier}\rangle$  should simply switch to the installed *math group* instead of calling `\select@group` unnecessarily. So we globally redefine the first argument (the new  $\langle\mathit{math alphabet identifier}\rangle$ ) to expand into a `\mathgroup` switch and then select this *alphabet*. Note that this redefinition will be overwritten by the next call to a *version* macro. The original code for the end of `\select@group` was

```
\gdef#1{#3\mathgroup #2}#1\fi}
```

i.e. first redefining the  $\langle\mathit{math alphabet identifier}\rangle$  and then calling the new definition to switch to the wanted  $\langle\mathit{math group}\rangle$ . Now we define the  $\langle\mathit{math alphabet identifier}\rangle$  as a call to the `\use@mathgroup` command.

```
595 % \xdef#1{\noexpand\use@mathgroup\noexpand#2%
596 % {\number\csname c@mv@\math@version\endcsname}}%
```

But this is not sufficient, as we learned the hard way. The problem here is that the loading of the fonts that comprise the alphabet identifier #1, as well as the necessary math font assignments is deferred until it is used. This is OK so far, but if the fonts are switched within the current formula (which may happen if a sub-formula is a box that contains a math version switch) the font assignments for #1 are not restored unless #1 is used again. This is disastrous since TeX sees the wrong fonts at the end of the math formula, when it converts the math list into a horizontal list.

This is taken into account as follows: When a math alphabet identifier is used for the first time in a certain version it modifies the corresponding macro `\mv@<version>` so that it calls `\getanddefine@fonts` directly in future as well. We use the macro `\extract@alph@from@version` to do this. It takes the math alphabet identifier #1 and the math version macro as arguments.

```
597 % \expandafter\extract@alph@from@version
598 % \csname mv@\math@version\expandafter\endcsname
599 % \expandafter{\number\csname c@mv@\math@version\endcsname}%
600 % #1%
601 % \stepcounter{mv@\math@version}%
```

Finally, it is not possible to simply call the new definition since we have an argument (the third argument of `\use@mathgroup` or more exactly the argument of `\math@egroup` if the `margid` option is in force) which would swallow our closing `\fi`. So we use the `\expandafter` technique to remove the `\fi` before the `\use@mathgroup` is expanded.

```
602 %\expandafter #1\fi
```

(End definition for `\select@group`.)

\extract@alph@from@version We proceed to the definition of the macro \extract@alph@from@version. As stated above, it takes a math alphabet identifier and a math version macro (e.g. \mv@normal) as its arguments.

603 \def\extract@alph@from@version#1#2#3{%

To extract and replace the definition of math alphabet identifier #3 in macro #1 we have to recall how this definition looks like: Somewhere in the replacement text of #1 there is the sequence

\install@mathalphabet<math alphabet identifier> #3{%  
(Definitions for )#3}

Hence, the first thing we do is to extract the tokens preceding this definitions, the definition itself, and the tokens following it. To this end we define one auxiliary macro \reserved@a.

604 \def\reserved@a##1\install@mathalphabet#3##2##3@nil{%

When \reserved@a is expanded, it will have the tokens preceding the definition in question in its first argument (#1), the following tokens in its third argument (#3), and the replacement text for the math alphabet identifier #3 in its second argument. (#2). This is then recorded for later use in a temporary macro \reserved@b.

605 \def\reserved@b{##2}{%

Additionally, we define a macro \reserved@c to reconstruct the definitions for the math version in question from the tokens that will remain unchanged (#1 and #3) and the yet to build new definitions for the math alphabet identifier #3.

606 \def\reserved@c####1{\gdef#1{##1####1##3}}{}

Then we execute our auxiliary macro.

607 \expandafter\reserved@a#1@nil

OK, so now we have to build the new definition for #3. To do so, we first extract the interesting parts out of the old one. The old definition looks like:

\select@group<math alphabet identifier>  
(math group number)<math extra part>  
<curr@fontshape definition>

So we define a new temporary macro \reserved@a that extracts these parts.

608 \def\reserved@a\select@group#3##1##2@nil{%

This macro can now directly rebuild the math version definition by calling \reserved@c:

609 \reserved@c{  
610 \getanddefine@fonts{#2}##2%  
611 \install@mathalphabet#3{  
612 \relax\ifmmode \else \non@alpherr#3\fi  
613 \use@mathgroup##1{#2}}}%

In addition it defines the alphabet the way it should be used from now on.

614 \gdef#3{\relax\ifmmode \else \non@alpherr#3\fi  
615 \use@mathgroup##1{#2}}{}

Finally, we only have to call this macro \reserved@a on the old definitions recorded in \reserved@b:

616 \expandafter\reserved@a\reserved@b@nil  
617 }

(End definition for \extract@alph@from@version.)

\math@bgroup Here are the default definitions for \math@bgroup and \math@egroup. We use \bgroup instead of \begingroup to avoid ‘leaking out’ of style changes. This has the side effect of always producing mathord atoms.

```
618 \let\math@bgroup\bgroup
619 \def\math@egroup#1{#1\egroup}
(End definition for \math@bgroup and \math@egroup.)
```

\calculate@math@sizes Here is the default definition for \calculate@math@sizes a more elaborate interface is under testing in mthscale.sty.

```
620 \gdef\calculate@math@sizes{%
621 \@font@info{Calculating space math\space sizes\space for\space
622 size\space <\f@size>}%
623 \dimen@\f@size \p@
624 \tempdima \defaultscriptratio \dimen@
625 \dimen@ \defaultscriptscriptratio \dimen@
626 \expandafter\xdef\csname S@\f@size\endcsname{%
627 \gdef\noexpand\sf@size{\f@size}%
628 \gdef\noexpand\ssf@size{\strip@pt\tempdima}%
629 \gdef\noexpand\sssf@size{\strip@pt\dimen@}%
630 \noexpand\math@fonttrue}}}
```

(End definition for \calculate@math@sizes.)

\defaultscriptratio The default ratio for math sizes is:

\defaultscriptscriptratio 1 to \defaultscriptratio to \defaultscriptscriptratio.  
By default this is 1 to .7 to .5.

```
631 \def\defaultscriptratio{.7}
632 \def\defaultscriptscriptratio{.5}
```

(End definition for \defaultscriptratio and \defaultscriptscriptratio.)

\noaccents@ If we don’t have a definition for \noaccents@ we provide a dummy.

```
633 \ifx\noaccents@\undefined
634 \let\noaccents@\empty
635 \fi
```

(End definition for \noaccents@.)

\showhyphens The \showhyphens command must be redefined since the version in plain.tex uses \tenrm. We have also made some further adjustments for its use in L<sup>A</sup>T<sub>E</sub>X.

```
636 //2ekernel%
637 <latexrelease>\IncludeInRelease[2017/01/01]{\showhyphens}%
638 <latexrelease> {XeTeX support for \showhyphens}%
639 <2ekernel | latexrelease>
640 \ifx\XeTeXcharclass\undefined
```

Version for engines other than XeT<sub>E</sub>X.

```
641 \DeclareRobustCommand\showhyphens[1]{%
642 \setbox0\vbox{%
643 \color@begingroup
644 \everypar{}%
645 \parfillskip\z@skip\hspace\maxdimen
646 \normalfont
647 \pretolerance\m@ne\tolerance\m@ne\hbadness\z@\showboxdepth\z@\ #1%
648 \color@endgroup}}
```

```
649 \else
```

Xe<sub>T</sub>E<sub>X</sub> version. When using system fonts Xe<sub>T</sub>E<sub>X</sub> reports consecutive runs of characters as a single item in box logging, which means the standard `\showhyphens` does not work. This version typesets the text into a narrow box to force hyphenation and then reconstructs a horizontal list with explicit hyphens to generate the display. Note that the `lmr` OpenType font is forced, this works even if the characters are not in the font as hyphenation is attempted due to the width of the space and hyphen character. It may generate spurious Missing Character warnings in the log, these are however suppressed from the terminal output by ensuring that `\tracingonline` is locally zero.

```
650 \DeclareRobustCommand\showhyphens[1]{%
651 \setbox0\vbox{%
652 \usefont{TU}{lmr}{m}{n}%
653 \hsize 1sp %
654 \hbadness\@M
655 \hfuzz\maxdimen
656 \tracingonline\z@%
657 \everypar={}\%
658 \leftskip\z@skip
659 \rightskip\z@skip
660 \parfillskip\z@skip
661 \hyphenpenalty=-\@M
662 \pretolerance\m@ne
663 \interlinepenalty\z@%
664 \clubpenalty\z@%
665 \widowpenalty\z@%
666 \brokenpenalty1127 %
667 \setbox\z@\hbox{}%
668 \noindent
669 \hskip\z@skip
670 #1%
671 \par}
```

Note here we stop the loop if made no progress, non-removable items may mean that we can not process the whole list (which would be testable as `\lastnodetype=-1`).

```
672 \loop
673 \if@tempswafalse
674 \ifnum\lastnodetype=11\unskip\@tempswatrue\fi
675 \ifnum\lastnodetype=12\unkern\@tempswatrue\fi
676 \ifnum\lastnodetype=13 %
677 \count@\lastpenalty
678 \unpenalty\@tempswatrue
679 \fi
680 \ifnum\lastnodetype=\@ne
681 \setbox\tw@\lastbox\@tempswatrue
682 \setbox0\hbox{\unhbox\tw@\unskip\unskip\unpenalty
683 \ifnum\count@=1127 \else\ \fi
684 \unhbox0}%
685 \count@\z@
686 \fi
687 \if@tempswa
688 \repeat
689 \hbadness\z@
690 \hsize\maxdimen
```

```

691 \showboxdepth\z@

692 \tolerance\m@ne

693 \hyphenpenalty\z@

694 \noindent\unhbox\z@

695 }}

696 \fi

697 </2ekernel | latexrelease>

698 <| latexrelease>\EndIncludeInRelease

699 <| latexrelease>\IncludeInRelease{0000/00/00}{\showhyphens} %

700 <| latexrelease> {XeTeX support for \showhyphens} %

701 <| latexrelease>\gdef\showhyphens#1{ %

702 <| latexrelease> \setbox0\vbox{ %

703 <| latexrelease> \color@begingroup %

704 <| latexrelease> \everypar{} %

705 <| latexrelease> \parfillskip\z@skip\hspace\maxdimen

706 <| latexrelease> \normalfont

707 <| latexrelease> \pretolerance\m@ne\tolerance\m@ne

708 <| latexrelease> \hbadness\z@\showboxdepth\z@\ #1%

709 <| latexrelease> \color@endgroup} }

710 <| latexrelease>\EndIncludeInRelease

711 <*2ekernel>

```

(End definition for `\showhyphens`.)

`\addto@hook` We need a macro to add tokens to a hook.

```
712 \long\def\addto@hook#1#2{\expandafter{\the#1#2}}
```

(End definition for `\addto@hook`.)

`\@vpt`

```
713 \def\@vpt{5}
```

(End definition for `\@vpt`.)

`\@vipt`

```
714 \def\@vipt{6}
```

(End definition for `\@vipt`.)

`\@viipt`

```
715 \def\@viipt{7}
```

(End definition for `\@viipt`.)

`\@viiiip`

```
716 \def\@viiiip{8}
```

(End definition for `\@viiiip`.)

`\@ixpt`

```
717 \def\@ixpt{9}
```

(End definition for `\@ixpt`.)

```

\@xpt
718 \def\@xpt{10}
(End definition for \@xpt.)

\@xipt
719 \def\@xipt{10.95}
(End definition for \@xipt.)

\@xiipt
720 \def\@xiipt{12}
(End definition for \@xiipt.)

\@xivpt
721 \def\@xivpt{14.4}
(End definition for \@xivpt.)

\@xviipt
722 \def\@xviipt{17.28}
(End definition for \@xviipt.)

\@xxpt
723 \def\@xxpt{20.74}
(End definition for \@xxpt.)

\@xxvpt
724 \def\@xxvpt{24.88}
(End definition for \@xxvpt.)
725 ⟨/2ekernel⟩

```

## File w

# ltfssaxes.dtx

This file contains the implementation for handling extra axes splitting the series and the values into sub-categories. selection commands. See other parts of the L<sup>A</sup>T<sub>E</sub>X distribution, or *The L<sup>A</sup>T<sub>E</sub>X Companion* for higher level documentation of the L<sup>A</sup>T<sub>E</sub>X Font Selection Scheme.

Everything in the this file got introduced 2020/02/02, so we use large rollback chunks, only interrupted if necessary.

```
1 <*2ekernel | latexrelease>
2 <latexrelease>\IncludeInRelease{2020/02/02}%
3 <latexrelease> {\DeclareFontSeriesChangeRule}{Series change rules}%
```

## 1 Changing the font series

In the original NFSS implementation the series was a single attribute stored in `\f@series` and so one always had to specify both weight and width together. This means it was impossible to typeset, a paragraph in a condensed font and inside have a few words in bold weight (but still condensed) without doing this manually by requesting `\fontseries{bc}\selectfont`.

The new implementation now works differently by looking both at the current value of `\f@series` and the requested new series and out of that combination selects a resulting series value. Thus, if the current series is `c` and we ask for `b` we now get `bc`.

This is done by consulting a simple lookup table. This table is configurable (though most likely that flexibility will seldom of ever be needed) Adding or changing entries in this table are done with `\DeclareFontSeriesChangeRule`.

### 1.1 The series lookup table

`\DeclareFontSeriesChangeRule` The `\DeclareFontSeriesChangeRule` defines entries in a simple database (implemented as a set of commands) that define mappings between from an existing series and requested new series and maps that to a result series (and additionally offers an alternative if the desired one is not existing):

```
#1 current \f@series
#2 requested new series
#3 result (if that exist for the given font family)
#4 alternative result (if #3 does not exist)
```

If an `.fd` file has its own substitution rules then #3 exist and thus #4 is not applied.

If there is no matching database entry or if neither the result nor the alternate result exist in the font family the requested new series is used (which then may trigger substitutions later on).

```
4 \def\DeclareFontSeriesChangeRule#1#2#3#4{%
5 \namedef{series@#1@#2}{\{\!\{#3\!\}{\!\!} {#4\!\}}}}
```

(End definition for `\DeclareFontSeriesChangeRule`.)

## 1.2 Mapping rules for series changes

The rules set up use explicit series values not `\..default` indirections; my current feeling is that this is in fact better.

With 9 weights and 9 width classes this table is getting a bit large in the end (324 entries) but on the other hand it doesn't change and accessing speed and it is fast this way.

We could alternatively split the axis and maintain weight and width separately, but that would take more processing time and would not allow for setting up explicit exceptions nicely (not sure that this would ever get used though).

Design considerations for mapping entries:

- We make `m` to reset both weight and width (as this is how it always worked). To reset just the width `?m` is provided and to reset just the weight `m?`.
- We do support “`mwidth`” and “`weightm`”, e.g., `mec` to mean “go to medium weight and extra-condensed width”. At the end of the process we automatically drop any leftover `m` in the series name (unless it is just a single `m`).
- If there is no table entry then the target series is used unconditionally. This means that any request to set both weight and width (e.g. `bx` or `ulc`) needs no table entries. For that reason there are no entries which have a weight+width as request (i.e., second argument).

In particular this is also true for cases involving `m`, e.g., `bm` (bold medium width) which automatically gets reduced result in `b` or `mc` (medium weight condensed) which becomes `c` as a result.

- Only a few entries have “alternative” values and perhaps most of them should get dropped. Or maybe not ... needs some thought perhaps.

The idea is that you don't want the normal substitution to kick in because that would reset the shape first and it may be better to stay with `b` when a change to `c` is requested and `bc` doesn't exist, than to go to first change the shape to `n` and then find that `bc/n` doesn't exist either and thus ending up with `m/n`.

- Also: while I did set up all nine standard weight values from `ul` to `ub` I only bothered to provide entries for `ec`, `sc`, `c` and `x`, because other levels of compression/expansion are not in any real fonts that I know.

Could and perhaps should be eventually extended to cover the whole set.

```
6 \DeclareFontSeriesChangeRule {bc}{b}{bc}{}
7 \DeclareFontSeriesChangeRule {bc}{c}{bc}{}
8 \DeclareFontSeriesChangeRule {bc}{eb}{ebc}{}
9 \DeclareFontSeriesChangeRule {bc}{ec}{bec} {bc}
10 \DeclareFontSeriesChangeRule {bc}{el}{elc}{}
11 \DeclareFontSeriesChangeRule {bc}{l}{lc}{}
12 \DeclareFontSeriesChangeRule {bc}{sb}{sbc}{}
13 \DeclareFontSeriesChangeRule {bc}{sc}{bsc} {bc}
14 \DeclareFontSeriesChangeRule {bc}{s1}{slc}{}
15 \DeclareFontSeriesChangeRule {bc}{ub}{ubc}{}
16 \DeclareFontSeriesChangeRule {bc}{ul}{ulc}{}
17 \DeclareFontSeriesChangeRule {bc}{x}{bx}{}
18 \end{document}
```

```

18 \DeclareFontSeriesChangeRule {bx}{b}{bx}{}%
19 \DeclareFontSeriesChangeRule {bx}{c} {bc} {bx} %<-----
20 \DeclareFontSeriesChangeRule {bx}{eb}{ebx}{}%
21 \DeclareFontSeriesChangeRule {bx}{ec} {bec} {bx} %<-----
22 \DeclareFontSeriesChangeRule {bx}{el}{elx}{}%
23 \DeclareFontSeriesChangeRule {bx}{l}{lx}{}%
24 \DeclareFontSeriesChangeRule {bx}{sb} {sbx} {}%
25 \DeclareFontSeriesChangeRule {bx}{sc} {bsc} {bx} %<-----
26 \DeclareFontSeriesChangeRule {bx}{sl}{slx} {}%
27 \DeclareFontSeriesChangeRule {bx}{ub}{ubx}{}%
28 \DeclareFontSeriesChangeRule {bx}{ul}{ulx}{}%
29 \DeclareFontSeriesChangeRule {bx}{x}{bx}{}%

30 \DeclareFontSeriesChangeRule {b}{bx} {bx} {b} %<-----
31 \DeclareFontSeriesChangeRule {b}{c} {bc} {b} %<-----
32 \DeclareFontSeriesChangeRule {b}{ec} {bec} {b} %<-----
33 \DeclareFontSeriesChangeRule {b}{sb} {sb} {b} %<-----
34 \DeclareFontSeriesChangeRule {b}{sc} {bsc} {b} %<-----
35 \DeclareFontSeriesChangeRule {b}{x} {bx} {b} %<-----

36 \DeclareFontSeriesChangeRule {c}{bx} {bx} {b} %<-----
37 \DeclareFontSeriesChangeRule {c}{b}{bc}{}%
38 \DeclareFontSeriesChangeRule {c}{eb}{ebc}{}%
39 \DeclareFontSeriesChangeRule {c}{el}{elc}{}%
40 \DeclareFontSeriesChangeRule {c}{l}{lc}{}%
41 \DeclareFontSeriesChangeRule {c}{sb}{sbc}{}%
42 \DeclareFontSeriesChangeRule {c}{sl}{slc}{}%
43 \DeclareFontSeriesChangeRule {c}{ub}{ubc}{}%
44 \DeclareFontSeriesChangeRule {c}{ul}{ulc}{}%
45 \DeclareFontSeriesChangeRule {c}{x}{x}{m} %<-----

46 \DeclareFontSeriesChangeRule {ebc}{b}{bc}{}%
47 \DeclareFontSeriesChangeRule {ebc}{c}{ebc}{}%
48 \DeclareFontSeriesChangeRule {ebc}{eb}{ebc}{}%
49 \DeclareFontSeriesChangeRule {ebc}{ec}{ebec}{ebc}%
50 \DeclareFontSeriesChangeRule {ebc}{el}{elc}{}%
51 \DeclareFontSeriesChangeRule {ebc}{l}{lc}{}%
52 \DeclareFontSeriesChangeRule {ebc}{sb}{sbc}{}%
53 \DeclareFontSeriesChangeRule {ebc}{sc}{ebsc}{ebc}%
54 \DeclareFontSeriesChangeRule {ebc}{sl}{slc}{}%
55 \DeclareFontSeriesChangeRule {ebc}{ub}{ubc}{}%
56 \DeclareFontSeriesChangeRule {ebc}{ul}{ulc}{}%
57 \DeclareFontSeriesChangeRule {ebc}{x}{ebx}{}%

58 \DeclareFontSeriesChangeRule {ec}{bx} {bx} {b} %<-----
59 \DeclareFontSeriesChangeRule {ec}{b}{bec}{}%
60 \DeclareFontSeriesChangeRule {ec}{eb}{ebc}{}%
61 \DeclareFontSeriesChangeRule {ec}{el}{elec}{}%
62 \DeclareFontSeriesChangeRule {ec}{l}{lec}{}%
63 \DeclareFontSeriesChangeRule {ec}{sb}{sbec}{}%
64 \DeclareFontSeriesChangeRule {ec}{sl}{slec}{}%
65 \DeclareFontSeriesChangeRule {ec}{ub}{ubec}{}%
66 \DeclareFontSeriesChangeRule {ec}{ul}{ulec}{}%
67 \DeclareFontSeriesChangeRule {ec}{x}{x}{m} %<-----

68 \DeclareFontSeriesChangeRule {sc}{bx} {bx} {b} %<-----
69 \DeclareFontSeriesChangeRule {sc}{b}{bsc}{}%

```

```

70 \DeclareFontSeriesChangeRule {sc}{eb}{ebsc}{}
71 \DeclareFontSeriesChangeRule {sc}{el}{elsc}{}
72 \DeclareFontSeriesChangeRule {sc}{l}{lsc}{}
73 \DeclareFontSeriesChangeRule {sc}{sb}{sbsc}{}
74 \DeclareFontSeriesChangeRule {sc}{s1}{slsc}{}
75 \DeclareFontSeriesChangeRule {sc}{ub}{ubsc}{}
76 \DeclareFontSeriesChangeRule {sc}{ul}{ulsc}{}
77 \DeclareFontSeriesChangeRule {sc}{x}{x}{m} %<-----

78 \DeclareFontSeriesChangeRule {ebx}{b}{bx}{}
79 \DeclareFontSeriesChangeRule {ebx}{c}{ebc}{}
80 \DeclareFontSeriesChangeRule {ebx}{eb}{ebx}{}
81 \DeclareFontSeriesChangeRule {ebx}{ec}{ebec}{}
82 \DeclareFontSeriesChangeRule {ebx}{el}{elx}{}
83 \DeclareFontSeriesChangeRule {ebx}{l}{lx}{}
84 \DeclareFontSeriesChangeRule {ebx}{sb}{sbx}{}
85 \DeclareFontSeriesChangeRule {ebx}{sc}{ebsc}{}
86 \DeclareFontSeriesChangeRule {ebx}{s1}{slx}{}
87 \DeclareFontSeriesChangeRule {ebx}{ub}{ubx}{}
88 \DeclareFontSeriesChangeRule {ebx}{ul}{ulx}{}
89 \DeclareFontSeriesChangeRule {ebx}{x}{ebx}{}

90 \DeclareFontSeriesChangeRule {eb}{c}{ebc}{}
91 \DeclareFontSeriesChangeRule {eb}{ec}{ebec}{}
92 \DeclareFontSeriesChangeRule {eb}{sc}{ebsc}{}
93 \DeclareFontSeriesChangeRule {eb}{x}{ebx}{}

94 \DeclareFontSeriesChangeRule {elc}{b}{bc}{}
95 \DeclareFontSeriesChangeRule {elc}{c}{elc}{}
96 \DeclareFontSeriesChangeRule {elc}{eb}{ebc}{}
97 \DeclareFontSeriesChangeRule {elc}{ec}{elec}{}
98 \DeclareFontSeriesChangeRule {elc}{el}{elc}{}
99 \DeclareFontSeriesChangeRule {elc}{l}{lc}{}
100 \DeclareFontSeriesChangeRule {elc}{sb}{sbc}{}
101 \DeclareFontSeriesChangeRule {elc}{sc}{elsc}{}
102 \DeclareFontSeriesChangeRule {elc}{s1}{slc}{}
103 \DeclareFontSeriesChangeRule {elc}{ub}{ubc}{}
104 \DeclareFontSeriesChangeRule {elc}{ul}{ulc}{}
105 \DeclareFontSeriesChangeRule {elc}{x}{elx}{}

106 \DeclareFontSeriesChangeRule {elx}{b}{bx}{}
107 \DeclareFontSeriesChangeRule {elx}{c}{elc}{}
108 \DeclareFontSeriesChangeRule {elx}{eb}{ebx}{}
109 \DeclareFontSeriesChangeRule {elx}{ec}{elec}{}
110 \DeclareFontSeriesChangeRule {elx}{el}{elx}{}
111 \DeclareFontSeriesChangeRule {elx}{l}{lx}{}
112 \DeclareFontSeriesChangeRule {elx}{sb}{sbx}{}
113 \DeclareFontSeriesChangeRule {elx}{sc}{elsc}{}
114 \DeclareFontSeriesChangeRule {elx}{s1}{slx}{}
115 \DeclareFontSeriesChangeRule {elx}{ub}{ubx}{}
116 \DeclareFontSeriesChangeRule {elx}{ul}{ulx}{}
117 \DeclareFontSeriesChangeRule {elx}{x}{elx}{}

118 \DeclareFontSeriesChangeRule {el}{c}{elc}{}
119 \DeclareFontSeriesChangeRule {el}{ec}{elec}{}
120 \DeclareFontSeriesChangeRule {el}{sc}{elsc}{}
121 \DeclareFontSeriesChangeRule {el}{x}{elx}{}

```

```

122 \DeclareFontSeriesChangeRule {lc}{b}{bc}={}
123 \DeclareFontSeriesChangeRule {lc}{c}{lc}={}
124 \DeclareFontSeriesChangeRule {lc}{eb}{ebc}={}
125 \DeclareFontSeriesChangeRule {lc}{ec}{lec}={}
126 \DeclareFontSeriesChangeRule {lc}{el}{elc}={}
127 \DeclareFontSeriesChangeRule {lc}{l}{lc}={}
128 \DeclareFontSeriesChangeRule {lc}{sb}{sbc}={}
129 \DeclareFontSeriesChangeRule {lc}{sc}{lsc}={}
130 \DeclareFontSeriesChangeRule {lc}{s1}{slc}={}
131 \DeclareFontSeriesChangeRule {lc}{ub}{ubc}={}
132 \DeclareFontSeriesChangeRule {lc}{ul}{ulc}={}
133 \DeclareFontSeriesChangeRule {lc}{x}{lx}={}

134 \DeclareFontSeriesChangeRule {lx}{b}{bx}={}
135 \DeclareFontSeriesChangeRule {lx}{c}{lc}={}
136 \DeclareFontSeriesChangeRule {lx}{eb}{ebx}={}
137 \DeclareFontSeriesChangeRule {lx}{ec}{lec}={}
138 \DeclareFontSeriesChangeRule {lx}{el}{elx}={}
139 \DeclareFontSeriesChangeRule {lx}{l}{lx}={}
140 \DeclareFontSeriesChangeRule {lx}{sb}{sbx}={}
141 \DeclareFontSeriesChangeRule {lx}{sc}{lsc}={}
142 \DeclareFontSeriesChangeRule {lx}{s1}{slx}={}
143 \DeclareFontSeriesChangeRule {lx}{ub}{ubx}={}
144 \DeclareFontSeriesChangeRule {lx}{ul}{ulx}={}
145 \DeclareFontSeriesChangeRule {lx}{x}{lx}={}

146 \DeclareFontSeriesChangeRule {l}{bx}{bx}{b} %<-----
147 \DeclareFontSeriesChangeRule {l}{b}{b}{bx} %<-----
148 \DeclareFontSeriesChangeRule {l}{c}{lc}{l} % ? %<-----
149 \DeclareFontSeriesChangeRule {l}{ec}{lec}{l} % ? %<-----
150 \DeclareFontSeriesChangeRule {l}{sb}{sb}{b} % ? %<-----
151 \DeclareFontSeriesChangeRule {l}{sc}{lsc}{l} % ? %<-----
152 \DeclareFontSeriesChangeRule {l}{x}{lx}{l} % ? %<-----

153 \DeclareFontSeriesChangeRule {m}{bx}{bx}{b} %<-----
154 \DeclareFontSeriesChangeRule {m}{b}{b}{bx} %<-----
155 \DeclareFontSeriesChangeRule {m}{c}{c}{m} %<-----
156 \DeclareFontSeriesChangeRule {m}{ec}{ec}{m} %<-----
157 \DeclareFontSeriesChangeRule {m}{l}{l}{m} %<-----
158 \DeclareFontSeriesChangeRule {m}{sb}{sb}{b} %<-----
159 \DeclareFontSeriesChangeRule {m}{sc}{sc}{m} %<-----
160 \DeclareFontSeriesChangeRule {m}{x}{x}{m} %<-----

161 \DeclareFontSeriesChangeRule {sbc}{b}{bc}={}
162 \DeclareFontSeriesChangeRule {sbc}{c}{sbc}={}
163 \DeclareFontSeriesChangeRule {sbc}{eb}{ebc}={}
164 \DeclareFontSeriesChangeRule {sbc}{ec}{sbec}{sbc}
165 \DeclareFontSeriesChangeRule {sbc}{el}{elc}={}
166 \DeclareFontSeriesChangeRule {sbc}{l}{lc}={}
167 \DeclareFontSeriesChangeRule {sbc}{sb}{sbc}={}
168 \DeclareFontSeriesChangeRule {sbc}{sc}{sbsc}{sbc}
169 \DeclareFontSeriesChangeRule {sbc}{s1}{slc}={}
170 \DeclareFontSeriesChangeRule {sbc}{ub}{ubc}={}
171 \DeclareFontSeriesChangeRule {sbc}{ul}{ulc}={}
172 \DeclareFontSeriesChangeRule {sbc}{x}{sbx}={}

173 \DeclareFontSeriesChangeRule {sbx}{b}{bx}={}

```

```

174 \DeclareFontSeriesChangeRule {sbx}{c}{sbc}{}

175 \DeclareFontSeriesChangeRule {sbx}{eb}{ebx}{}

176 \DeclareFontSeriesChangeRule {sbx}{ec}{sbec}{}

177 \DeclareFontSeriesChangeRule {sbx}{el}{elx}{}

178 \DeclareFontSeriesChangeRule {sbx}{l}{lx}{}

179 \DeclareFontSeriesChangeRule {sbx}{sb}{sbx}{}

180 \DeclareFontSeriesChangeRule {sbx}{sc}{sbsc}{}

181 \DeclareFontSeriesChangeRule {sbx}{s1}{slx}{}

182 \DeclareFontSeriesChangeRule {sbx}{ub}{ubx}{}

183 \DeclareFontSeriesChangeRule {sbx}{ul}{ulx}{}

184 \DeclareFontSeriesChangeRule {sbx}{x}{sbx}{}

185 \DeclareFontSeriesChangeRule {sb}{c} {sbc} {bc} %? %<----

186 \DeclareFontSeriesChangeRule {sb}{ec} {sbec} {sbc} %? %<----

187 \DeclareFontSeriesChangeRule {sb}{sc} {sbsc} {sbc} %? %<----

188 \DeclareFontSeriesChangeRule {sb}{x} {sbx} {bx} %? %<----

189 \DeclareFontSeriesChangeRule {slc}{b}{bc}{}

190 \DeclareFontSeriesChangeRule {slc}{c}{slc}{}

191 \DeclareFontSeriesChangeRule {slc}{eb}{ebc}{}

192 \DeclareFontSeriesChangeRule {slc}{ec}{slec}{}

193 \DeclareFontSeriesChangeRule {slc}{el}{elc}{}

194 \DeclareFontSeriesChangeRule {slc}{l}{lc}{}

195 \DeclareFontSeriesChangeRule {slc}{sb}{sbc}{}

196 \DeclareFontSeriesChangeRule {slc}{sc}{slsc}{}

197 \DeclareFontSeriesChangeRule {slc}{s1}{slc}{}

198 \DeclareFontSeriesChangeRule {slc}{ub}{ubc}{}

199 \DeclareFontSeriesChangeRule {slc}{ul}{ulc}{}

200 \DeclareFontSeriesChangeRule {slc}{x}{slx}{}

201 \DeclareFontSeriesChangeRule {slx}{b}{bx}{}

202 \DeclareFontSeriesChangeRule {slx}{c}{slc}{}

203 \DeclareFontSeriesChangeRule {slx}{eb}{ebx}{}

204 \DeclareFontSeriesChangeRule {slx}{ec}{slec}{}

205 \DeclareFontSeriesChangeRule {slx}{el}{elx}{}

206 \DeclareFontSeriesChangeRule {slx}{l}{lx}{}

207 \DeclareFontSeriesChangeRule {slx}{sb}{sbc}{}

208 \DeclareFontSeriesChangeRule {slx}{sc}{slsc}{}

209 \DeclareFontSeriesChangeRule {slx}{s1}{slx}{}

210 \DeclareFontSeriesChangeRule {slx}{ub}{ubx}{}

211 \DeclareFontSeriesChangeRule {slx}{ul}{ulx}{}

212 \DeclareFontSeriesChangeRule {slx}{x}{slx}{}

213 \DeclareFontSeriesChangeRule {s1}{c}{slc}{}

214 \DeclareFontSeriesChangeRule {s1}{ec}{slec}{}

215 \DeclareFontSeriesChangeRule {s1}{sc}{slsc}{}

216 \DeclareFontSeriesChangeRule {s1}{x}{slx}{}

217 \DeclareFontSeriesChangeRule {ubc}{b}{bc}{}

218 \DeclareFontSeriesChangeRule {ubc}{c}{ubc}{}

219 \DeclareFontSeriesChangeRule {ubc}{eb}{ebc}{}

220 \DeclareFontSeriesChangeRule {ubc}{ec}{ubec}{}

221 \DeclareFontSeriesChangeRule {ubc}{el}{elc}{}

222 \DeclareFontSeriesChangeRule {ubc}{l}{lc}{}

223 \DeclareFontSeriesChangeRule {ubc}{sb}{sbc}{}

224 \DeclareFontSeriesChangeRule {ubc}{sc}{ubsc}{}

225 \DeclareFontSeriesChangeRule {ubc}{s1}{slc}{}


```

```

226 \DeclareFontSeriesChangeRule {ubc}{ub}{ubc}{}
227 \DeclareFontSeriesChangeRule {ubc}{ul}{ulc}{}
228 \DeclareFontSeriesChangeRule {ubc}{x}{ubx}{}
229 \DeclareFontSeriesChangeRule {ubx}{b}{bx}{}
230 \DeclareFontSeriesChangeRule {ubx}{c}{ubc}{}
231 \DeclareFontSeriesChangeRule {ubx}{eb}{ebx}{}
232 \DeclareFontSeriesChangeRule {ubx}{ec}{ubec}{}
233 \DeclareFontSeriesChangeRule {ubx}{el}{elx}{}
234 \DeclareFontSeriesChangeRule {ubx}{l}{lx}{}
235 \DeclareFontSeriesChangeRule {ubx}{sb}{sbx}{}
236 \DeclareFontSeriesChangeRule {ubx}{sc}{ubsc}{}
237 \DeclareFontSeriesChangeRule {ubx}{s1}{slx}{}
238 \DeclareFontSeriesChangeRule {ubx}{ub}{ubx}{}
239 \DeclareFontSeriesChangeRule {ubx}{ul}{ulx}{}
240 \DeclareFontSeriesChangeRule {ubx}{x}{ubx}{}
241 \DeclareFontSeriesChangeRule {ub}{c}{ubc}{}
242 \DeclareFontSeriesChangeRule {ub}{ec}{ubec}{}
243 \DeclareFontSeriesChangeRule {ub}{sc}{ubsc}{}
244 \DeclareFontSeriesChangeRule {ub}{x}{ubx}{}
245 \DeclareFontSeriesChangeRule {ulc}{b}{bc}{}
246 \DeclareFontSeriesChangeRule {ulc}{c}{ulc}{}
247 \DeclareFontSeriesChangeRule {ulc}{eb}{ebc}{}
248 \DeclareFontSeriesChangeRule {ulc}{ec}{ulec}{ulc}
249 \DeclareFontSeriesChangeRule {ulc}{el}{elc}{}
250 \DeclareFontSeriesChangeRule {ulc}{l}{lc}{}
251 \DeclareFontSeriesChangeRule {ulc}{sb}{sbc}{}
252 \DeclareFontSeriesChangeRule {ulc}{sc}{ulsc}{ulc}
253 \DeclareFontSeriesChangeRule {ulc}{s1}{slc}{}
254 \DeclareFontSeriesChangeRule {ulc}{ub}{ubc}{}
255 \DeclareFontSeriesChangeRule {ulc}{ul}{ulc}{}
256 \DeclareFontSeriesChangeRule {ulc}{x}{ulx}{}
257 \DeclareFontSeriesChangeRule {ulx}{b}{bx}{}
258 \DeclareFontSeriesChangeRule {ulx}{c}{ulc}{}
259 \DeclareFontSeriesChangeRule {ulx}{eb}{ebx}{}
260 \DeclareFontSeriesChangeRule {ulx}{ec}{ulec}{}
261 \DeclareFontSeriesChangeRule {ulx}{el}{elx}{}
262 \DeclareFontSeriesChangeRule {ulx}{l}{lx}{}
263 \DeclareFontSeriesChangeRule {ulx}{sb}{sbx}{}
264 \DeclareFontSeriesChangeRule {ulx}{sc}{ulsc}{}
265 \DeclareFontSeriesChangeRule {ulx}{s1}{slx}{}
266 \DeclareFontSeriesChangeRule {ulx}{ub}{ubx}{}
267 \DeclareFontSeriesChangeRule {ulx}{ul}{ulx}{}
268 \DeclareFontSeriesChangeRule {ulx}{x}{ulx}{}
269 \DeclareFontSeriesChangeRule {ul}{c}{ulc}{}
270 \DeclareFontSeriesChangeRule {ul}{ec}{ulec}{}
271 \DeclareFontSeriesChangeRule {ul}{sc}{ulsc}{}
272 \DeclareFontSeriesChangeRule {ul}{x}{ulx}{}
273 \DeclareFontSeriesChangeRule {x}{b}{bx}{}
274 \DeclareFontSeriesChangeRule {x}{c}{c}{}
275 \DeclareFontSeriesChangeRule {x}{eb}{ebx}{}
276 \DeclareFontSeriesChangeRule {x}{ec}{ec}{}
277 \DeclareFontSeriesChangeRule {x}{el}{elx}{}

```

```

278 \DeclareFontSeriesChangeRule {x}{l}{lx}{}
279 \DeclareFontSeriesChangeRule {x}{sb}{sbx}{}
280 \DeclareFontSeriesChangeRule {x}{sc}{sc}{}
281 \DeclareFontSeriesChangeRule {x}{sl}{slx}{}
282 \DeclareFontSeriesChangeRule {x}{ub}{ubx}{}
283 \DeclareFontSeriesChangeRule {x}{ul}{ulx}{}

```

Special rules for `lm` etc. aren't needed because if the target `lm` is requested it will be used if there is no rule and that is then reduced to `l` automatically. Same for `mc` and friends. Only `?m` and `m?` need rules.

So here are the special rules for `m?`:

```

284 \DeclareFontSeriesChangeRule {bc}{m?}{c}{}
285 \DeclareFontSeriesChangeRule {bec}{m?}{ec}{}
286 \DeclareFontSeriesChangeRule {bsc}{m?}{sc}{}
287 \DeclareFontSeriesChangeRule {bx}{m?}{x}{}
288 \DeclareFontSeriesChangeRule {b}{m?}{m}{}
289 \DeclareFontSeriesChangeRule {c}{m?}{c}{}
290 \DeclareFontSeriesChangeRule {ebc}{m?}{c}{}
291 \DeclareFontSeriesChangeRule {ebec}{m?}{ec}{}
292 \DeclareFontSeriesChangeRule {ebsc}{m?}{sc}{}
293 \DeclareFontSeriesChangeRule {ebx}{m?}{x}{}
294 \DeclareFontSeriesChangeRule {eb}{m?}{m}{}
295 \DeclareFontSeriesChangeRule {ec}{m?}{ec}{}
296 \DeclareFontSeriesChangeRule {elc}{m?}{c}{}
297 \DeclareFontSeriesChangeRule {elec}{m?}{ec}{}
298 \DeclareFontSeriesChangeRule {elsc}{m?}{sc}{}
299 \DeclareFontSeriesChangeRule {elx}{m?}{x}{}
300 \DeclareFontSeriesChangeRule {el}{m?}{m}{}
301 \DeclareFontSeriesChangeRule {lc}{m?}{c}{}
302 \DeclareFontSeriesChangeRule {lec}{m?}{ec}{}
303 \DeclareFontSeriesChangeRule {lsc}{m?}{sc}{}
304 \DeclareFontSeriesChangeRule {lx}{m?}{x}{}
305 \DeclareFontSeriesChangeRule {l}{m?}{m}{}
306 \DeclareFontSeriesChangeRule {m}{m?}{m}{}
307 \DeclareFontSeriesChangeRule {sbc}{m?}{c}{}
308 \DeclareFontSeriesChangeRule {sbec}{m?}{ec}{}
309 \DeclareFontSeriesChangeRule {sbsc}{m?}{sc}{}
310 \DeclareFontSeriesChangeRule {sbx}{m?}{x}{}
311 \DeclareFontSeriesChangeRule {sb}{m?}{m}{}
312 \DeclareFontSeriesChangeRule {sc}{m?}{sc}{}
313 \DeclareFontSeriesChangeRule {slc}{m?}{c}{}
314 \DeclareFontSeriesChangeRule {slec}{m?}{ec}{}
315 \DeclareFontSeriesChangeRule {slsc}{m?}{sc}{}
316 \DeclareFontSeriesChangeRule {slx}{m?}{x}{}
317 \DeclareFontSeriesChangeRule {sl}{m?}{m}{}
318 \DeclareFontSeriesChangeRule {ubc}{m?}{c}{}
319 \DeclareFontSeriesChangeRule {ubec}{m?}{ec}{}
320 \DeclareFontSeriesChangeRule {ubsc}{m?}{sc}{}
321 \DeclareFontSeriesChangeRule {ubx}{m?}{x}{}
322 \DeclareFontSeriesChangeRule {ub}{m?}{ub}{}
323 \DeclareFontSeriesChangeRule {ulc}{m?}{c}{}
324 \DeclareFontSeriesChangeRule {ulec}{m?}{ec}{}
325 \DeclareFontSeriesChangeRule {ulsc}{m?}{sc}{}
326 \DeclareFontSeriesChangeRule {ulx}{m?}{x}{}

```

```

327 \DeclareFontSeriesChangeRule {ul}{m?}{m}{}
328 \DeclareFontSeriesChangeRule {x}{m?}{x}{}
 And there the special rules for ?m:
329 \DeclareFontSeriesChangeRule {bc}{?m}{b}{}
330 \DeclareFontSeriesChangeRule {bec}{?m}{b}{}
331 \DeclareFontSeriesChangeRule {bsc}{?m}{b}{}
332 \DeclareFontSeriesChangeRule {bsc}{?m}{b}{}
333 \DeclareFontSeriesChangeRule {bx}{?m}{b}{}
334 \DeclareFontSeriesChangeRule {b}{?m}{b}{}
335 \DeclareFontSeriesChangeRule {c}{?m}{m}{}
336 \DeclareFontSeriesChangeRule {ebc}{?m}{eb}{}
337 \DeclareFontSeriesChangeRule {ebec}{?m}{eb}{}
338 \DeclareFontSeriesChangeRule {ebsc}{?m}{eb}{}
339 \DeclareFontSeriesChangeRule {ebsc}{?m}{eb}{}
340 \DeclareFontSeriesChangeRule {ebx}{?m}{eb}{}
341 \DeclareFontSeriesChangeRule {eb}{?m}{eb}{}
342 \DeclareFontSeriesChangeRule {ec}{?m}{m}{}
343 \DeclareFontSeriesChangeRule {elc}{?m}{el}{}
344 \DeclareFontSeriesChangeRule {elec}{?m}{el}{}
345 \DeclareFontSeriesChangeRule {elsc}{?m}{el}{}
346 \DeclareFontSeriesChangeRule {elsc}{?m}{el}{}
347 \DeclareFontSeriesChangeRule {elx}{?m}{el}{}
348 \DeclareFontSeriesChangeRule {el}{?m}{el}{}
349 \DeclareFontSeriesChangeRule {lc}{?m}{l}{}
350 \DeclareFontSeriesChangeRule {lec}{?m}{l}{}
351 \DeclareFontSeriesChangeRule {lsc}{?m}{l}{}
352 \DeclareFontSeriesChangeRule {lsc}{?m}{l}{}
353 \DeclareFontSeriesChangeRule {lx}{?m}{l}{}
354 \DeclareFontSeriesChangeRule {l}{?m}{l}{}
355 \DeclareFontSeriesChangeRule {m}{?m}{m}{}
356 \DeclareFontSeriesChangeRule {sbc}{?m}{sb}{}
357 \DeclareFontSeriesChangeRule {sbec}{?m}{sb}{}
358 \DeclareFontSeriesChangeRule {sbsc}{?m}{sb}{}
359 \DeclareFontSeriesChangeRule {sbsc}{?m}{sb}{}
360 \DeclareFontSeriesChangeRule {sbx}{?m}{sb}{}
361 \DeclareFontSeriesChangeRule {sb}{?m}{sb}{}
362 \DeclareFontSeriesChangeRule {sc}{?m}{m}{}
363 \DeclareFontSeriesChangeRule {sc}{?m}{m}{}
364 \DeclareFontSeriesChangeRule {slc}{?m}{sl}{}
365 \DeclareFontSeriesChangeRule {slecl}{?m}{sl}{}
366 \DeclareFontSeriesChangeRule {s1sc}{?m}{sl}{}
367 \DeclareFontSeriesChangeRule {s1sc}{?m}{sl}{}
368 \DeclareFontSeriesChangeRule {slx}{?m}{sl}{}
369 \DeclareFontSeriesChangeRule {sl}{?m}{sl}{}
370 \DeclareFontSeriesChangeRule {ubc}{?m}{ub}{}
371 \DeclareFontSeriesChangeRule {ubec}{?m}{ub}{}
372 \DeclareFontSeriesChangeRule {ubsc}{?m}{ub}{}
373 \DeclareFontSeriesChangeRule {ubsc}{?m}{ub}{}
374 \DeclareFontSeriesChangeRule {ubx}{?m}{ub}{}
375 \DeclareFontSeriesChangeRule {ub}{?m}{m}{}
376 \DeclareFontSeriesChangeRule {ulc}{?m}{ul}{}
377 \DeclareFontSeriesChangeRule {ulec}{?m}{ul}{}
378 \DeclareFontSeriesChangeRule {ulsc}{?m}{ul}{}
379 \DeclareFontSeriesChangeRule {ulsc}{?m}{ul}{}

```

```

380 \DeclareFontSeriesChangeRule {ulx}{?m}{ul}{}
381 \DeclareFontSeriesChangeRule {ul}{?m}{ul}{}
382 \DeclareFontSeriesChangeRule {x}{?m}{m}{}
 Supporting rollback ...
383 </2ekernel | latexrelease>
384 <latexrelease>\EndIncludeInRelease
385 <latexrelease>\IncludeInRelease{0000/00/00}%
386 <latexrelease> {\DeclareFontSeriesChangeRule}{Series change rules}%
387 <latexrelease>
388 <latexrelease>\let\DeclareFontSeriesChangeRule\@undefined
389 <latexrelease>
390 <latexrelease>\EndIncludeInRelease

```

### 1.3 Changing to a new series

```

391 <*2ekernel | latexrelease>
392 <latexrelease>\IncludeInRelease{2021/06/01}%
393 <latexrelease> {\fontseries}{delay fontseries update}%

```

**\fontseries** The `\fontseries` command takes one argument which is the requested new font series. In the orginal implementation it simply saved the expanded value in `\f@series`. Now we do a bit more processing and look up the final value in the font series data base. This is done by `\merge@font@series`. But the lookup should be done within the target family and call to `\fontseries` might be followed by a `\fontfamily` call. So we delay the processing to `\selectfont` and only record the necessary action in `\delayed@f@adjustment`.

```

394 \DeclareRobustCommand\fontseries[1]{\@forced@seriesfalse
395 \expandafter\def\expandafter\delayed@f@adjustment\expandafter
396 {\delayed@f@adjustment\delayed@merge@font@series{\#1}}}

```

(*End definition for \fontseries.*)

**\delayed@f@adjustment** The macro holding the delayed action(s) for use in `\selectfont`.

```

397 \let\delayed@f@adjustment\empty

```

(*End definition for \delayed@f@adjustment.*)

**\fontseriesforce** To change unconditionally to a new series you can use `\fontseriesforce`. Of course, if the series doesn't exist for the current family substitution still happens, but there is not dependency on the current series.

```

398 \DeclareRobustCommand\fontseriesforce[1]{\@forced@seriestrue
399 \expandafter\def\expandafter\delayed@f@adjustment\expandafter
400 {\delayed@f@adjustment\edef\f@series{\#1}}}

```

(*End definition for \fontseriesforce.*)

**\if@forced@series** If the series gets forced we need to know that fact later on.

```

401 \newif\if@forced@series

```

(*End definition for \if@forced@series.*)

```

402 </2ekernel | latexrelease>
403 <latexrelease>\EndIncludeInRelease

```

```

404 〈latexrelease〉\IncludeInRelease{2020/02/02}%
405 〈latexrelease〉 {\fontseries}{delay fontseries update}%
406 〈latexrelease〉
407 〈latexrelease〉\DeclareRobustCommand\fontseries[1]{\@forced@seriesfalse
408 〈latexrelease〉 \merge@font@series{\#1}%
409 〈latexrelease〉\DeclareRobustCommand\fontseriesforce[1]{\@forced@seriestrue
410 〈latexrelease〉 \edef\f@series{\#1}%
411 〈latexrelease〉\let\delayed@f@adjustment\@undefined
412 〈latexrelease〉

```

For a roll forward we may have to define `\if@forced@series` but this needs doing in a way that `\TeX` doesn't see it when skipping over conditionals.

```

413 〈latexrelease〉\expandafter\newif\csname if@forced@series\endcsname
414 〈latexrelease〉
415 〈latexrelease〉\EndIncludeInRelease

416 〈latexrelease〉\IncludeInRelease{0000/00/00}%
417 〈latexrelease〉 {\fontseries}{delay fontseries update}%
418 〈latexrelease〉
419 〈latexrelease〉\DeclareRobustCommand\fontseries[1]{\edef\f@series{\#1}}
420 〈latexrelease〉\let\fontseriesforce\@undefined
421 〈latexrelease〉
422 〈latexrelease〉\EndIncludeInRelease

423 〈*2ekernel | latexrelease〉
424 〈latexrelease〉\IncludeInRelease{2020/02/02}%
425 〈latexrelease〉 {\merge@font@series}{Merge series values}%

```

`\merge@font@series` We look up the data base value by expanding the right command twice. If no such value exist then the result will be `\relax` otherwise it will be the two brace groups: the desired result and the alternate result. The first case means that the third argument to `\merge@font@series` will be empty.

```

426 \def\merge@font@series#1{%
427 \expandafter\expandafter\expandafter
428 \merge@font@series@
429 \csname series@\f@series \#1\endcsname
430 {\#1}%
431 \nil
432 }

```

(End definition for `\merge@font@series`.)

`\merge@font@series@` This now defines the new `\f@series`:

```
433 \def\merge@font@series@#1#2#3\@nil{%
```

If the third argument is empty there is no database entry for the combination and the second argument holds the new series so we return that.

Originally the test was simply `\ifx!#3!` but that actually dies if #3 starts with a conditional and in the definition of `\AmSfont` that is actually the case.

```

434 \%ifcat\expandafter X\detokenize{\#1}X%
435 \def\reserved@a{\#3}%
436 \ifx\reserved@a\empty
437 \set@target@series{\#2}%
438 \else

```

Otherwise we check if the desired result for the series (#1) exists for the font family and the current shape. All this happens inside `\selectfont` which has already taken care to load the `.fd` file if necessary.

```
439 \edef\reserved@a{\f@encoding / \f@family / #1 / \f@shape}%
440 \ifcsname \reserved@a \endcsname
```

If the desired result is available then we use that. However, we do need some post-processing because we need to drop surplus `ms` due to the way naming convention was designed in the '90s (sigh).

```
441 \set@target@series{#1}%
```

If not, then we try the alternate result (#2).

```
442 \else
443 \ifcsname \f@encoding / \f@family / #2 / \f@shape \endcsname
```

If the alternate result exist we use that and also issue a warning (or rather a log entry) that we didn't managed to change to the desired font.

```
444 \set@target@series{#2}%
445 \font@shape@subst@warning
```

If that doesn't exist either, then we use the requested series unmodified (again with a warning).

```
446 \else
447 \set@target@series{#3}%
448 \font@shape@subst@warning
449 \fi
450 \fi
451 \fi
452 }
```

It is possible that the previous font and the new one are actually identical (and the font was not found because it still needs loading) in which case a warning would look rather odd. So we make a quick check for that (which is the reason why we defined `\@reserved@a` above instead of doing inline testing inside `\ifcsname`).

```
453 \def\font@shape@subst@warning{%
454 \edef\reserved@b{\curr@fontshape}%
455 \ifx\reserved@a\reserved@b \else
456 \font@warning{Font shape '\reserved@a' undefined\MessageBreak
457 using '\reserved@b' instead}%
458 \fi
459 }
```

*(End definition for `\merge@font@series`.)*

```
\merge@font@series@without@substitution
\merge@font@series@without@substitution@
\delayed@merge@font@series
```

`\merge@font@series@without@substitution` works like `\merge@font@series`, i.e., it looks up the combination in the rule base and if there exists an entry it uses it and if not it uses the new series value. However, it doesn't check if there is actually a font face with the new series value as `\merge@font@series` does. This simplified command is used in `\selectfont` at a point where other font attributes are not yet updated so that checking the font face might result incorrect in substitutions.

```
460 \def\merge@font@series@without@substitution#1{%
461 \expandafter\expandafter\expandafter
462 \merge@font@series@without@substitution@
463 \csname series@\f@series @#1\endcsname
```

```

464 {#1}%
465 \@nil
466 }
467 \def\merge@font@series@without@substitution{\@nil{%
468 \def\reserved@a{#3}%
469 \ifx\reserved@a\empty
470 \set@target@series{#2}%
471 \else
472 \set@target@series{#1}%
473 \fi
474 }

(End definition for \merge@font@series@without@substitution,
 \merge@font@series@without@substitution@, and \delayed@merge@font@series.)
```

`\delayed@merge@font@series` When we delay the merge action in `\fontseries` we first attempt to use merging without substitution. If that results in a non-existing font face the merge is redone in `\selectfont` using a version with substitution. See `\selectfont` for details.

```

475 \let\delayed@merge@font@series\merge@font@series@without@substitution
476
(End definition for \delayed@merge@font@series.)
```

`\maybe@load@fontshape` A small helper that we use a couple of times: try loading a fontshape (in a group because `\try@load@fontshape` normalizes catcodes and we also want to change `\typeout` so that it doesn't report missing .fd files on the terminal).

```

476 \def\maybe@load@fontshape{%
477 \begingroup
478 \let \typeout \font@info
479 \try@load@fontshape
480 \endgroup}
```

(End definition for `\maybe@load@fontshape`.)

`\set@target@series` Finally the code for normalizing the `\f@series` value.

The combined series value determined by the mapping may still contain an `m` that we have to remove (as the .fd files use `c` not `mc` to denote a medium weight condensed series, etc.). We do this in all branches above because a user might have written

```
\DeclareFontSeriesChangeRule {m}{sc}{msc}{mc}
```

instead of using `sc` and `c` as needed in the .fd file.

```
481 \def\set@target@series#1{%
```

We need to `\edef` the argument first in case it starts with a conditional. Then we check (and perhaps drop) an "m" from the value and assign the result to `\f@series`.

```

482 \edef\f@series{#1}%
483 \series@maybe@drop@one@m\f@series\f@series
484 }
```

(End definition for `\set@target@series`.)

\series@maybe@drop@one@m If the series value is in NFSS notation then it should not contain any “m” unless it is just an “m” by its own. So we need to drop surplus “m”s. But we better don’t do this for full names, such as “semibold” as used by `autoinst`, for example. So we test against the possible explicit values that should drop an “m”. After that we assign the result to #2 for further use.

```
485 \def\series@maybe@drop@one@m#1{%
486 \expandafter\series@maybe@drop@one@m@\expandafter{\#1}}%
487
488 \def\series@maybe@drop@one@m@x#1#2{%
```

The code below is an inline version of the `\in@` macro without the group, so that it works in `\accent`.

```
489 \def\in@##1,#1,{}%
490 \series@check@toks\expandafter{\in@%
491 ,ulm,elm,lm,slm,mm,sbm,bm,ebm,ubm,muc,mec,mc,msc,msx,mx,mex,mux,{},{} ,#1,}%
492 \edef\in@{\the\series@check@toks}%
493 \ifx\in@{\empty}
```

The default definition for `\bfdefault` etc is actually `b\empty` so that we can detect if the user has changed the default. However that means a) the above test will definitely fail (maybe something to change) and b) we better use `\edef` on the next line to get rid of it as otherwise the test against #2 (e.g. `\bfdef@ult`) will fail in other places.

```
494 \edef#2{\#1}%
495 \else
496 \edef#2{\expandafter\series@drop@one@m #1\series@drop@one@m}%
497 \fi
498 }
```

As a precaution we use a private toks register not `\toks@` as that is no longer hidden inside the group.

```
499 \newtoks\series@check@toks
(End definition for \series@maybe@drop@one@m.)
```

\series@drop@one@m Drop up to two `ms` but keep one if that makes the series value empty. Actually, with the current implementation we know that there is at least one in the series value itself and we added one after it, so all we have to do is now returning #1#2 and dropping the rest.

```
500 \def\series@drop@one@m#1#2m#3\series@drop@one@m{%
501 % \ifx\relax#1#2\relax m\else#1#2\fi
502 #1#2%
503 }
```

```
(End definition for \series@drop@one@m.)
Supporting rollback ...
504 </2ekernel | latexrelease>
505 <latexrelease>\EndIncludeInRelease
506 <latexrelease>\IncludeInRelease{0000/00/00}%
507 <latexrelease> {\merge@font@series}{Merge series values}%
508 <latexrelease>
509 <latexrelease>\let\merge@font@series@\undefined
510 <latexrelease>\let\merge@font@series@\undefined
511 <latexrelease>\let@font@shape@subst@warning@\undefined
512 <latexrelease>\let\merge@font@series@without@substitution@\undefined
513 <latexrelease>\let\merge@font@series@without@substitution@\undefined
```

```

514 〈latexrelease〉\let\delayed@merge@font@series\@undefined
515 〈latexrelease〉\let\maybe@load@fontshape\@undefined
516 〈latexrelease〉\let\set@target@series\@undefined
517 〈latexrelease〉\let\series@maybe@drop@one@m\@undefined
518 〈latexrelease〉\let\series@drop@one@m\@undefined
519 〈latexrelease〉
520 〈latexrelease〉\EndIncludeInRelease

```

## 2 Changing the shape

Shapes are also split in two axes (though it could be more if that is desirable), essentially building in an “sc” axis).

```

521 {*2ekernel | latexrelease}
522 〈latexrelease〉\IncludeInRelease{2020/02/02}%
523 〈latexrelease〉 {\DeclareFontShapeChangeRule}{Font shape change rules}%

```

`\DeclareFontShapeChangeRule` The database for shapes is done in exactly the same way, only that it is much smaller and we usually have no alternative shape (or rather it is empty thus not used).

```

524 \def\DeclareFontShapeChangeRule #1#2#3#4{%
525 \@namedef{shape@#1@#2}{\{\!#3\!\}{\!#4\!\}}}

```

(*End definition for \DeclareFontShapeChangeRule.*)

There is kind of the same problem with returning back from `sc` to normal. It sort of needs its own letter. In `fontspec` this was solved by the first time `\upshape` changes `it` or `sl` back (so only `sc` remains) and second time it changes then `sc` back to normal. Maybe that’s not a bad way to handle it, but decided for a slightly different approach: `n` always returns to “normal”, ie resets everything and `up` changes italic or slanted to upright and `ulc` undoes small caps.

So we now offer `\normalshape` (using `\shapedefault` which is normally the same as calling both `\ulcshape` and `\upshape`, only more efficient.

`\ulcshape` To request going back to upper/lowercase we need a new command. It uses `ulc` as shape name but this shape is virtual, i.e., it doesn’t exist as a real shape, it is only used as part of the database table entries and thus only appears in the second argument there (but not in the first).

```

526 \DeclareRobustCommand\ulcshape
527 {\not@math@alphabet\ulcshape\relax
528 \fontshape\ulcdefault\selectfont}
529 \let\ulcdefault\@undefined % for rollback
530 \newcommand\ulcdefault{ulc}

```

(*End definition for \ulcshape, \textulc, and \ulcdefault.*)

`\swshape` New command to select a swash shape. The standard rules put this in the same category as italics or slanted, i.e., if you ask for it then italics are undone. One could provide more complicated rules so that `it + sw` becomes `swit` but given that there are only very few fonts that have swash letters that level of flexibility (these days) would be just resulting in a lot of combinations that do not exist.

```

531 \DeclareRobustCommand\swshape
532 {\not@math@alphabet\swshape\relax
533 \fontshape\swdefault\selectfont}
534 \let\swdefault\@undefined % for rollback
535 \newcommand\swdefault{sw}

```

(End definition for `\swshape`, `\textsw`, and `\swdefault`.)

- `\sscshape` New command to select spaced small capitals. This is only here because `fontaxes` offered it. There isn't a single free font that supports it. However, some commercial ones do, so we offer it so that at some point `fontaxes` could be retired.

So far there aren't any rules for it—probably there should be some putting it in the same category as `sc`.

```
536 \DeclareRobustCommand\sscshape
537 {\not@math@alphabet\sscshape\relax
538 \fontshape\sscdefault\selectfont}
539 \let\sscdefault\@undefined % for rollback
540 \newcommand\sscdefault{sc}
```

(End definition for `\sscshape`, `\textssc`, and `\sscdefault`.)

## 2.1 Mapping rules for shape combinations

Many of the entries are commented out as we will get that result without any entry.

```
541 \%{\DeclareFontShapeChangeRule {n}{n} {n} {}}
542 \DeclareFontShapeChangeRule {n}{it} {it} {sl}
543 \DeclareFontShapeChangeRule {n}{sl} {sl} {it}
544 \%{\DeclareFontShapeChangeRule {n}{sw} {sw} {}}
545 \%{\DeclareFontShapeChangeRule {n}{sc} {sc} {}}
546 \DeclareFontShapeChangeRule {n}{ulc} {n} {}
547 \DeclareFontShapeChangeRule {n}{up} {n} {}
548 \%{\DeclareFontShapeChangeRule {it}{n} {n} {}}
549 \%{\DeclareFontShapeChangeRule {it}{it} {it} {}}
550 \DeclareFontShapeChangeRule {it}{sl} {sl} {it}
551 \%{\DeclareFontShapeChangeRule {it}{sw} {sw} {}}
```

If neither `scit` nor `scls` exist then `sc` will be used as a fallback albeit with a log entry, so except for the latter there will be no change for CM or Latin Modern fonts.

```
552 \DeclareFontShapeChangeRule {it}{sc} {scit} {scls}
553 \DeclareFontShapeChangeRule {it}{ulc} {it} {}
554 \DeclareFontShapeChangeRule {it}{up} {n} {}
555 \%{\DeclareFontShapeChangeRule {sl}{n} {n} {}}
556 \DeclareFontShapeChangeRule {sl}{it} {it} {sl}
557 \%{\DeclareFontShapeChangeRule {sl}{sl} {sl} {}}
558 \%{\DeclareFontShapeChangeRule {sl}{sw} {sw} {}}
559 \DeclareFontShapeChangeRule {sl}{sc} {scls} {scit}
560 \DeclareFontShapeChangeRule {sl}{ulc} {sl} {}
561 \DeclareFontShapeChangeRule {sl}{up} {n} {}
562 \%{\DeclareFontShapeChangeRule {sc}{n} {n} {}}
563 \DeclareFontShapeChangeRule {sc}{it} {scit} {scls}
564 \DeclareFontShapeChangeRule {sc}{sl} {scls} {scit}
565 \DeclareFontShapeChangeRule {sc}{sw} {scsw} {sw}
566 \%{\DeclareFontShapeChangeRule {sc}{sc} {sc} {}}
567 \DeclareFontShapeChangeRule {sc}{ulc} {n} {}
```

The next rule might be a bit surprising and rightly so. Correct would be that `sc` is not affected by `up`, i.e., remains `sc` as showed in the commented out rule. However, for nearly three decades commands such as `sc` or `\textup` changed small caps back to the “normal” shape. So for backward compatibility we keep hat behavior.

As a result you are currently typesetting in `scit` or `scls` using `\upshape` twice will return you to the normal shape too, the first will change to `sc` and the second (because of the rule below) change that to `n`. This is the way `fontspec` implemented its version on this interface, so this rule means we are also compatible with the way `fontspec` behaved. Still it remains an oddity which I would rather liked to have avoided.

```

568 \%{\! DeclareFontShapeChangeRule {sc}{up} {sc} {}}
569 \DeclareFontShapeChangeRule {sc}{up} {n} {}
570 \%{\! DeclareFontShapeChangeRule {scit}{n} {n} {}}
571 \DeclareFontShapeChangeRule {scit}{it} {scit} {}
572 \DeclareFontShapeChangeRule {scit}{sl} {scls} {scit}
573 \DeclareFontShapeChangeRule {scit}{sw} {scsw} {sc} % or scit?
574 \DeclareFontShapeChangeRule {scit}{sc} {scit} {}
575 \DeclareFontShapeChangeRule {scit}{ulc} {it} {}
576 \DeclareFontShapeChangeRule {scit}{up} {sc} {}

```

The previous rule assumes that if `scit` exists then `it` exists as well. If not, the mechanism will save `ulc` in `\f@series` which most certainly doesn't exist. So when a font is later selected that would result in a substitution (so no harm done really). Alternatively, we could in this case use `n` as alternative, which may be a bit faster, but such a setup would be so weird in the first place that this isn't worth the effort.

```

577 \%{\! DeclareFontShapeChangeRule {scls}{n} {n} {}}
578 \DeclareFontShapeChangeRule {scls}{it} {scit} {scls}
579 \DeclareFontShapeChangeRule {scls}{sl} {scls} {}
580 \DeclareFontShapeChangeRule {scls}{sw} {scsw} {sc} % or scls?
581 \DeclareFontShapeChangeRule {scls}{sc} {scls} {}
582 \DeclareFontShapeChangeRule {scls}{ulc} {sl} {}
583 \DeclareFontShapeChangeRule {scls}{up} {sc} {}

584 \%{\! DeclareFontShapeChangeRule {scsw}{n} {n} {}}
585 \DeclareFontShapeChangeRule {scsw}{it} {scit} {scsw}
586 \DeclareFontShapeChangeRule {scsw}{sl} {scls} {}
587 \DeclareFontShapeChangeRule {scsw}{sw} {scsw} {}
588 \DeclareFontShapeChangeRule {scsw}{sc} {scsw} {}
589 \DeclareFontShapeChangeRule {scsw}{ulc} {sw} {}
590 \DeclareFontShapeChangeRule {scsw}{up} {sc} {}

591 \%{\! DeclareFontShapeChangeRule {sw}{n} {n} {}}
592 \%{\! DeclareFontShapeChangeRule {sw}{it} {it} {}}
593 \%{\! DeclareFontShapeChangeRule {sw}{sl} {sl} {}}
594 \%{\! DeclareFontShapeChangeRule {sw}{sw} {sw} {}}
595 \DeclareFontShapeChangeRule {sw}{sc} {scsw} {}
596 \DeclareFontShapeChangeRule {sw}{ulc} {sw} {}
597 \DeclareFontShapeChangeRule {sw}{up} {n} {}

```

Supporting rollback ...

```

598 //2ekernel | latexrelease)
599 <latexrelease>\EndIncludeInRelease
600 <latexrelease>\IncludeInRelease{0000/00/00}%
601 <latexrelease> {\! DeclareFontShapeChangeRule}{Font shape change rules}%
602 <latexrelease>
603 <latexrelease>\let\DeclareFontShapeChangeRule\@undefined
604 <latexrelease>\let\ulcshape\@undefined
605 <latexrelease>\let\ulcdefault\@undefined
606 <latexrelease>\let\swshape\@undefined

```

```

607 〈\latexrelease〉\let\swdefault\@undefined
608 〈\latexrelease〉\let\sscshape\@undefined
609 〈\latexrelease〉\let\sscdefault\@undefined
610 〈\latexrelease〉
611 〈\latexrelease〉\EndIncludeInRelease

```

## 2.2 Changing to a new shape

```

612 〈*2ekernel | \latexrelease〉
613 〈\latexrelease〉\IncludeInRelease{2021/06/01}%
614 〈\latexrelease〉{\fontshape}{Font shape change}%

```

**\fontshape** Again the `\fontshape` now has to do a lookup to get to its new value in `\f@shape`. The method is exactly the same as in `\fontseries`.

```

615 \DeclareRobustCommand\fontshape[1]
616 {\expandafter\def\expandafter\delayed@f@adjustment\expandafter
617 {\delayed@f@adjustment\delayed@merge@font@shape{\#1}}}

```

(*End definition for \fontshape.*)

**\fontshapeforce** The unconditional version:

```

618 \DeclareRobustCommand\fontshapeforce[1]
619 {\expandafter\def\expandafter\delayed@f@adjustment\expandafter
620 {\delayed@f@adjustment\edef\f@shape{\#1}}}

```

(*End definition for \fontshapeforce.*)

Supporting rollback ...

```

621 〈/2ekernel | \latexrelease〉
622 〈\latexrelease〉\EndIncludeInRelease
623 〈\latexrelease〉\IncludeInRelease{2020/02/02}%
624 〈\latexrelease〉{\fontshape}{Font shape change}%
625 〈\latexrelease〉
626 〈\latexrelease〉\DeclareRobustCommand\fontshape[1]{\merge@font@shape{\#1}}
627 〈\latexrelease〉\DeclareRobustCommand\fontshapeforce[1]{\edef\f@shape{\#1}}
628 〈\latexrelease〉
629 〈\latexrelease〉\EndIncludeInRelease
630 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
631 〈\latexrelease〉{\fontshape}{Font shape change}%
632 〈\latexrelease〉
633 〈\latexrelease〉\DeclareRobustCommand\fontshape [1]{\edef\f@shape{\#1}}
634 〈\latexrelease〉\let\fontshapeforce\@undefined
635 〈\latexrelease〉
636 〈\latexrelease〉\EndIncludeInRelease
637 〈*2ekernel | \latexrelease〉
638 〈\latexrelease〉\IncludeInRelease{2020/02/02}%
639 〈\latexrelease〉{\merge@font@shape}{Font shape change rules}%

```

**\merge@font@shape** Look up the database entry (if existing) and act accordingly.

```

640 \def\merge@font@shape#1{%
641 \expandafter\expandafter\expandafter
642 \merge@font@shape@
643 \csname shape@\f@shape @#1\endcsname
644 {#1}%

```

```

645 \cnil
646 }

```

(End definition for `\merge@font@shape.`)

`\merge@font@shape@` Same game now, except that we look at shapes not series values and we can set the shape without the complication of dropping “m”s from the name as we had to for the series.

```

647 \def\merge@font@shape@#1#2#3\cnil{%
648 \def\reserved@a{#3}%
649 \ifx\reserved@a\empty
650 \edef\f@shape{#2}%
651 \else

```

`\reserved@a` is used in `\@font@shape@subst@warning` so we have to define it in addition to do the `\ifcsname` test

```

652 \edef\reserved@a{\f@encoding / \f@family / \f@series/#1}%
653 \ifcsname \reserved@a\endcsname
654 \edef\f@shape{#1}%
655 \else
656 \ifcsname \f@encoding / \f@family / \f@series/#2\endcsname
657 \edef\f@shape{#2}%
658 \@font@shape@subst@warning
659 \else
660 \edef\f@shape{#3}%
661 \@font@shape@subst@warning
662 \fi
663 \fi
664 \fi
665 }

```

(End definition for `\merge@font@shape@.`)

`\merge@font@shape@without@substitution` See definition of `\selectfont` for how these macros are used.

```

666 \def\merge@font@shape@without@substitution#1{%
667 \expandafter\expandafter\expandafter
668 \merge@font@shape@without@substitution@
669 \csname shape@\f@shape \#1\endcsname
670 {#1}%
671 \cnil
672 }

```

```

673 \def\merge@font@shape@without@substitution@#1#2#3\cnil{%
674 \def\reserved@a{#3}%
675 \ifx\reserved@a\empty
676 \edef\f@shape{#2}%
677 \else
678 \edef\f@shape{#1}%
679 \fi
680 }

```

```

681 \let\delayed@merge@font@shape\merge@font@shape@without@substitution

```

(End definition for `\merge@font@shape@without@substitution`,  
`\merge@font@shape@without@substitution@`, and `\delayed@merge@font@shape.`)

```
\normalshape \normalshape resets both sub-axes if the default rules are used.

682 \protected\def\normalshape
683 {\not@math@alphabet\normalshape\relax
684 \fontshape\shapedefault\selectfont}%

(End definition for \normalshape.)
```

### 3 Make sure we win ...

This code implements one aspect of what the package `fontaxes` provide. So its redefinitions for the various shape commands, such as `\itshape` should no longer happen. We therefore force the standard definitions at `\AtBeginDocument` (later when this is defined. Once `fontaxes` is no longer doing such redefinitions that could be taken out again.

We use a separate macro so that we can easily disable this (in case of rollback).

`\reinstall@nfss@defs` I use `\protected` here not `\DeclareRobustCommand` to avoid extra status lines.

```
685 \def\reinstall@nfss@defs{%
686 \protected\def\upshape
687 {\not@math@alphabet\upshape\relax
688 \fontshape\updefault\selectfont}%
689 \protected\def\slshape
690 {\not@math@alphabet\slshape\relax
691 \fontshape\sldefault\selectfont}%
692 \protected\def\scshape
693 {\not@math@alphabet\scshape\relax
694 \fontshape\scdefault\selectfont}%
695 \protected\def\itshape
696 {\not@math@alphabet\itshape\mathit
697 \fontshape\itdefault\selectfont}%
698 \protected\def\ulcshape
699 {\not@math@alphabet\ulcshape\relax
700 \fontshape{ulc}\selectfont}%
701 \protected\def\swshape
702 {\not@math@alphabet\swshape\relax
703 \fontshape\swdefault\selectfont}%
704 \protected\def\sscshape
705 {\not@math@alphabet\sscshape\relax
706 \fontshape\sscdefault\selectfont}%
707 }
```

(End definition for `\reinstall@nfss@defs`.)

Supporting rollback ...

```
708 </2ekernel | latexrelease>
709 <latexrelease>\EndIncludeInRelease
710 <latexrelease>\IncludeInRelease{0000/00/00}%
711 <latexrelease> {\merge@font@shape}{Font shape change rules}%
712 <latexrelease>
713 <latexrelease>\DeclareRobustCommand\fontshape [1]{\edef\f@shape{\#1}}
714 <latexrelease>\let\fontshapeforce@\undefined
715 <latexrelease>
716 <latexrelease>\let\merge@font@shape@\undefined
717 <latexrelease>\let\merge@font@shape@@\undefined
718 <latexrelease>
```

```

719 〈\latexrelease〉\let\merge@font@shape@without@substitution@undefined
720 〈\latexrelease〉\let\merge@font@shape@without@substitution@\@undefined
721 〈\latexrelease〉\let\delayed@merge@font@shape@undefined
722 〈\latexrelease〉
723 〈\latexrelease〉\let\normalshape@undefined
724 〈\latexrelease〉

```

This is always called in \document so don't make it undefined.

```

725 〈\latexrelease〉
726 〈\latexrelease〉\let\reinstall@nfss@defs\relax
727 〈\latexrelease〉\EndIncludeInRelease

```

This initializes the 2020/02/02 extensions to NFSS after any changes in the preamble.

```

728 〈*2ekernel | \latexrelease〉
729 〈\latexrelease〉\IncludeInRelease{2020/10/01}%
730 〈\latexrelease〉
731 〈\g@addto@macro\@kernel@after@begindocument@before
732 〈\reinstall@nfss@defs\init@series@setup〉
733 〈/2ekernel | \latexrelease〉
734 〈\latexrelease〉\EndIncludeInRelease

```

The initialization was introduced in 2020/02/02 but

```

735 〈\latexrelease〉\IncludeInRelease{2020/02/02}%
736 〈\latexrelease〉
737 〈\latexrelease〉\AtBeginDocument{\reinstall@nfss@defs\init@series@setup}
738 〈\latexrelease〉\EndIncludeInRelease
739 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
740 〈\latexrelease〉
741 〈\latexrelease〉\EndIncludeInRelease
742 〈*2ekernel〉
743 〈/2ekernel〉

```

# File x

## ltfsstrc.dtx

### 1 Introduction

This package contains the code for tracing font loading and font changes. It basically overlays some of the low-level functions of NFSS with additional code used for tracing.

The package accepts the following options:

**errorshow** Write all information about font changes etc. only to the transcript file unless an error happens. This means that information about font substitution will not be shown on the terminal.

**warningshow** Show all NFSS warnings on the terminal. This setting corresponds to the default behaviour of NFSS if the **tracefnt** package is *not* loaded!

**infoshow** Show all NFSS warning and all NFSS info messages (that are normally only written to the transcript file) also on the terminal. This is the default if the **tracefnt** package is loaded.

**debugshow** In addition to **infoshow** show also changing of math fonts as far as possible (this option can produce a large amount of output).

**loading** Show the name of external fonts when they are loaded. This option shows only “newly” loaded fonts not those already preloaded in the format or the class file before the **tracefnt** package became active.

**pausing** Turn all font warnings into errors so that L<sup>A</sup>T<sub>E</sub>X will stop.

### 2 A driver for this document

The next bit of code contains the documentation driver file for T<sub>E</sub>X, i.e., the file that will produce the documentation you are currently reading. It will be extracted from this file by the DOCSTRIP program.

When this file is processed directly by L<sup>A</sup>T<sub>E</sub>X this will produce the documentation as well.

```
1 <*driver>
2 \documentclass{ltxdoc}
3
4
5 \%OnlyDescription % comment out for implementation details
6
7 \begin{document}
8 \DocInput{ltfsstrc.dtx}
9 \end{document}
10 </driver>
```

### 3 The Implementation

**Warning:** Read the macro documentation with a grain of salt. It is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

If we are making a package file it is a good idea to test whether we are running under 2e. This code is actually placed at the very beginning of this file for easier maintenance, thus commented out here.

```
11 <*package>
12 %\NeedsTeXFormat{LaTeX2e}
13 %\ProvidesPackage{tracefnt}[??/?/? v?.??
14 %
15 </package>
16 <+debug> \input trace.sty
```

The debug module makes use of commands contained in a special package file named `trace.sty`.<sup>25</sup>

### 4 Handling Options

\tracingfonts Here is the definition of the integer register for the font trace. As a default in a package file we use 1 to give error messages if fonts are substituted. If this code is used for debugging or tracing reasons in the format file (i.e. in `fam.dtx`) we use 0 as the default. But if no font trace is used we build a definition that will produce a warning message.

```
17 <*2ekernel>
18 \def\tracingfonts{%
19 \@font@warning{Command \noexpand\tracingfonts
20 not provided.\MessageBreak
21 Use the ‘tracefnt’ package.\MessageBreak Command found:}%
22 \count@}
23 </2ekernel>
```

The `\count@` in the line above will remove the number after `\tracingfonts`. Note that this definition will be overwritten by the next line if one of these modules are included.

```
24 <*package,trace,debug>
25 \newcount\tracingfonts
26 \tracingfonts=0
27 </package,trace,debug>
```

(End definition for `\tracingfonts`.)

The option `errorshow` turns off all warnings so that only real errors are shown. `warningshow` corresponds to the NFSS default (when `tracefnt` is not loaded). `infoshow` is the default for this package here; and `debugshow`, `loading`, and `pausing` extend the amount of information even further.

```
28 <*package>
29 \DeclareOption{errorshow}{%
30 \def\@font@info#1{%
31 \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
32 {LaTeX Font Info: \space\space\space#1}}%
```

---

<sup>25</sup>This package is not in distribution at the moment (and probably doesn't work any longer). Think of this part of the code as being historical artifacts.

```

33 \def\@font@warning#1{%
34 \GenericInfo{(Font)}{\spaces\spaces\space\space}%
35 {LaTeX Font Warning: #1}}%
36 }

37 \DeclareOption{warningshow}{%
38 \def\@font@info#1{%
39 \GenericInfo{(Font)}{\spaces\spaces\space\space}%
40 {LaTeX Font Info: \space\space\space#1}}%
41 \def\@font@warning#1{%
42 \GenericWarning{(Font)}{\spaces\space\space\space}%
43 {LaTeX Font Warning: #1}}%
44 }

45 \DeclareOption{infoshow}{%
46 \def\@font@info#1{%
47 \GenericWarning{(Font)}{\spaces\space\space\space}%
48 {LaTeX Font Info: \space\space\space#1}}%
49 \def\@font@warning#1{%
50 \GenericWarning{(Font)}{\spaces\space\space\space}%
51 {LaTeX Font Warning: #1}}%
52 }

53 \DeclareOption{loading}{%
54 \tracingfonts\tw@
55 }

56 \DeclareOption{debugshow}{%
57 \ExecuteOptions{infoshow}%
58 \tracingfonts\thr@@
59 }

60 \DeclareOption{pausing}{%
61 \def\@font@warning#1{%
62 \GenericError
63 {(Font)}{\spaces\space\space\space}%
64 {LaTeX Font Warning: #1}}%
65 {See the LaTeX Companion for details.}%
66 {I'll stop for every LaTeX Font Warning because
67 you requested\MessageBreak the 'pausing' option
68 to the tracefnt package.}}%
69 }

```

We make `infoshow` the default, which in turn defines `\font@warning` and `\font@info`.

```

70 \ExecuteOptions{infoshow}
71 \ProcessOptions
72
```

We also need a default definition inside the kernel:

```

73 <*2ekernel>
74 \def\@font@info#1{%
75 \GenericInfo{(Font)}{\spaces\space\space\space}%
76 {LaTeX Font Info: \space\space\space#1}}%
77 \def\@font@warning#1{%
78 \GenericWarning{(Font)}{\spaces\space\space\space}%
79 {LaTeX Font Warning: #1}}%
80
```

## 5 Macros common to `fam.tex` and `tracefnt.sty`

In the first versions of `tracefnt.dtx` some macros of `fam.dtx`<sup>26</sup> were redefined to included the extra tracing information. Now these macros are all defined in this file (i.e. removed from `fam.dtx`) and different production versions can be obtained simply by specifying a different set of modules to include when generating `ltfss.dtx`.

### 5.1 General font loading

- `\extract@font` This macro organizes the font loading. It first calls `\get@external@font` which will return in `\external@font` the name of the external font file (the `.tfm`) as it was determined by the NFSS tables.

```
81 {*2ekernel | package}
82 \def\extract@font{%
83 \get@external@font
```

Then the external font is loaded and assigned to the font identifier stored inside `\font@name` (for this reason we need `\expandafter`).

```
84 \global\expandafter\font\font@name\external@font\relax
```

When tracing we typeout the internal and external font name.

```
85 {*trace}
86 \ifnum \tracingfonts >@one
87 @font@info{External font '\external@font',
88 loaded as\MessageBreak \font@name}\fi
89
```

Finally we call the corresponding “loading action” macros to finish things. First the font is locally selected to allow the use of `\font` inside the loading action macros.

```
90 \font@name \relax
```

The next two lines execute the “loading actions” for the family and then for the individual font shape.

```
91 \csname \f@encoding+\f@family\endcsname
92 \csname\curr@fontshape\endcsname
93 \relax
94 }
95
```

The `\relax` at the end needs to be explained. This is inserted to prevent `TeX` from scanning too far when it is executing the replacement text of the loading code macros.

(*End definition for `\extract@font`.*)

- `\get@external@font` This function tries to find an external font name. It will place the name into the macro `\external@font`. If no font is found it will return the one that was defined via `\DeclareErrorFont`.

```
96 {*2ekernel}
97 \def\get@external@font{%
```

We don’t know the external font name at the beginning.

```
98 \let\external@font\@empty
99 \edef\font@info{\expandafter\expandafter\expandafter\string
100 \csname \curr@fontshape \endcsname}%
101 \try@size@range
```

---

<sup>26</sup>This file is currently not distributed in documented form. Its code is part of `ltfss.dtx`.

If this failed, we'll try to substitute another size of the same font. This is done by the `\try@size@substitution` macro. It "knows about" `\do@extract@font`, `\font@name`, `\f@size`, and so on.

```

102 \ifx\external@font\empty
103 \try@size@substitution
104 \ifx\external@font\empty
105 @latex@error{Font \expandafter \string\font@name\space
106 not found}\@eha
107 \error@fontshape
108 \get@external@font
109 \fi\fi
110 }
111
```

(End definition for `\get@external@font`.)

```

112 <*2ekernel | latexrelease | package>
113 <| latexrelease> \IncludeInRelease{2021/06/01}%
114 <| latexrelease> {\selectfont}{Add hook to \selectfont}%

```

`\selectfont` The macro `\selectfont` is called whenever a font change must take place.

```

115 \DeclareRobustCommand\selectfont
116 {%

```

When `debug` is specified we actually want something like 'undebbug'. The font selection is now stable so that using `\tracingall` on some other macros will show us a lot of unwanted information about font loading. Therefore we disable tracing during font loading as long as `\tracingfonts` is less than 4.

```

117 <+debug> \pushtracing
118 <+debug> \ifnum\tracingfonts<4 \tracingoff
119 <+debug> \else \tracingon\p@selectfont \fi

```

If `\baselinestretch` was redefined by the user it will not longer match its internal counterpart `\f@linespread`. If so we call `\set@fontsize` to prepare `\size@update`.

```

120 \ifx\f@linespread\baselinestretch \else
121 \set@fontsize\baselinestretch\f@size\f@baselineskip \fi

```

The series and shape updates are only prepared by `\fontseries` and `\fontshape` but not executed until after we are ready to change the font face. This way they happen after a possibly new family is set which is important because they look at the available font faces in that family and alter the selection based on availability. Several calls to `\fontseries` or `\fontshape` are delayed in the order in which they appear, so that by switching them one can work around missing intermediate font faces and avoid substitutions.

We first attempt to do the merge without any substitution. As we might end up with a non-existing font face we may have to restart and therefore save the current values of `\f@series` and `\f@shape` before the merge.

But first we make a quick test to see if there are any delayed actions, because if not it is pointless to make all the assignments and try loading a missing fontshape.

```

122 \ifx\delayed@f@adjustment\empty
123 \else
124 \let\f@shape@saved\f@shape
125 \let\f@series@saved\f@series

```

The we run the delayed adjustments (which is using the `\...@without@substitution` commands

```
126 \delayed@f@adjustment
```

We then check if the resulting combination is valid but for this we have to make sure that the appropriate .fd is loaded if that hasn't happened so far.

```
127 \maybe@load@fontshape
128 \ifcsname \f@encoding/\f@family/\f@series/\f@shape \endcsname
```

If this macro is defined then we are good and no further action is necessary.

Otherwise the combination is not valid, so we redo the merge but this time with substitutions.

```
129 \else
130 \let\f@shape\f@shape@saved
131 \let\f@series\f@series@saved
132 \let\delayed@merge@font@shape\merge@font@shape
133 \let\delayed@merge@font@series\merge@font@series
134 \delayed@f@adjustment
135 \let\delayed@merge@font@shape\merge@font@shape@without@substitution
136 \let\delayed@merge@font@series\merge@font@series@without@substitution
137 \fi
```

Now the series and shape values are updated and we clear `\delayed@f@adjustment`. This is important because on the next execution of `\selectfont` we should not mistakenly redo the delayed actions if there wasn't any series or shape change.

```
138 \let\delayed@f@adjustment\empty
139 \fi
```

If the series was forced we should now cancel that in case the next series change is done with some low-level setting to `\f@series`.

```
140 \forced@seriesfalse
```

Then we generate the internal name of the font by concatenating *family*, *series*, *shape*, and current *size*, with slashes as delimiters between them. This is much more readable than standard L<sup>A</sup>T<sub>E</sub>X's `\twfbf`, etc. We define `\font@name` globally, as always. The reason for this is explained later on.

```
141 \xdef\font@name{%
142 \csname\curr@fontshape/\f@size\endcsname}%
```

We call the macro `\pickup@font` which will load the font if necessary.

```
143 \pickup@font
```

Then we select the font.

```
144 \font@name
```

After switching fonts we run a hook, so that packages can make last minute alterations based on the new font (originally provided in `everysel` but using a different interface).

```
145 \UseHook{selectfont}%
```

Finally we call `\size@update`. This macro is normally empty but will contain actions (like setting the `\baselineskip`) that have to be carried out when the font size, the base `\baselineskip` or the `\baselinestretch` have changed.

```
146 \size@update
```

A similar function is called to handle anything related to encoding updates. This one is changed from `\relax` by `\fontencoding`.

```
147 \enc@update
```

Just before ending this macro we have to pop the tracing stack if it was pushed before.

```
148 <+debug> \poptracing
149 }
```

*(End definition for `\selectfont`.)*

**selectfont** Declare the hook used in `selecfont` in the kernel, but not inside the `tracefnt` package.

```
150 {-trace} \NewHook{selectfont}
```

*(End definition for `selectfont`.)*

If `\tracingfonts` is greater than 2 we also show the font switch inside `\selectfont`. We do this by adding this code to the hook in the `tracefnt` package: macro might redefine `\font@name`.

```
151 <*trace>
152 \AddToHook{selectfont}
153 { \ifnum \tracingfonts > \tw@
154 \font@info{Switching to \font@name}\fi
155 }
156 {/2ekernel | latexrelease | package}
157 \latexrelease\EndIncludeInRelease
```

With `\selectfont` having different definitions in different kernels we also have to provide them in the `tracefnt` package to support rollback. In packages that works a bit differently and therefore we have to provide an empty block there.

```
158 <package>\IncludeInRelease{2021/06/01}%
159 <package> {\selectfont}{Add hook to \selectfont}%
160 <package>\EndIncludeInRelease

161 <latexrelease | package>\IncludeInRelease{0000/00/00}%
162 <latexrelease | package> {\selectfont}{Add hook to \selectfont}%
163 <latexrelease | package>
164 <latexrelease | package>\DeclareRobustCommand\selectfont
165 <latexrelease | package> {%
166 <latexrelease | package> \ifx\f@linespread\baselinestretch \else
167 <latexrelease | package> \set@fontsize\baselinestretch\f@size\f@baselineskip \fi
168 <latexrelease | package> \xdef\font@name{%
169 \csname curr@fontshape/\f@size\endcsname}%
170 <latexrelease | package> \pickup@font
171 <latexrelease | package> \font@name
172 <latexrelease | package> \size@update
173 <latexrelease | package> \enc@update
174 <latexrelease | package> }
175 <latexrelease | package>
176 <latexrelease | package>\EndIncludeInRelease
```

**\set@fontsize** The macro `\set@fontsize` does the actual work. First it assigns new values to `\f@size`, `\f@baselineskip` and `\f@linespread`.

```
177 <*2ekernel | package>
178 \def\set@fontsize#1#2#3{%
179 \Q@defaultunits\Q@tempdimb#2pt\relax\Q@nnil
```

```

180 \edef\f@size{\strip@pt\@tempdimb}%
181 \@defaultunits\@tempskipa#3pt\relax\@nnil
182 \edef\f@baselineskip{\the\@tempskipa}%
183 \edef\f@linespread{\#1}%

```

For backward compatibility and for later testing within `\selectfont` the internal value of `\f@linespread` is passed back to `\baselinestretch`.

```
184 \let\baselinestretch\f@linespread
```

Additional processing will happen within `\selectfont`. For this reason the macro `\size@update` (which will be called in `\selectfont`) will be defined to be:

```
185 \def\size@update{%
```

First calculate the new `\baselineskip` and also store it in `normalbaselineskip`

```

186 \baselineskip\f@baselineskip\relax
187 \baselineskip\f@linespread\baselineskip
188 \normalbaselineskip\baselineskip

```

then to set up a new `\strutbox`

```

189 \setbox\strutbox\hbox{%
190 \vrule\@height.7\baselineskip
191 \@depth.3\baselineskip
192 \@width\z@}%

```

We end with a bit of tracing information.

```

193 <*>
194 \ifnum \tracingfonts>\tw@
195 \ifx\f@linespread\empty
196 \let\reserved@a\empty
197 \else
198 \def\reserved@a{\f@linespread x}%
199 \fi
200 \font@info{Changing size to \f@size/\reserved@a
201 \f@baselineskip}%
202 \aftergroup\type@restoreinfo \fi
203 </>

```

When all this is processed `\size@update` redefines itself to `\relax` so that in later calls of `\selectfont` no extra code will be executed.

```

204 \let\size@update\relax}%
205 }

```

Instead of defining this macro internally we might speed things up by placing the code into a separate macro and use `\let!`

*(End definition for `\set@fontsize`.)*

`\size@update` Normally this macro does nothing; it will be redefined by `\set@fontsize` to initiate an update.

```
206 \let\size@update\relax
```

*(End definition for `\size@update`.)*

`\type@restoreinfo` This macro produces some info when a font size and/or baseline change will get restored.

```
207 {*trace}
208 \def\type@restoreinfo{%
209 \ifx\f@linespread\empty
210 \let\reserved@a\empty
211 \else
212 \def\reserved@a{\f@linespread x}%
213 \fi
214 \font@info{Restoring size to
215 \f@size/\reserved@a\f@baselineskip}%
216 }
```

(End definition for `\type@restoreinfo`.)

`\glb@settings` The macro `\glb@settings` globally selects all math fonts for the current size if necessary.  
`\glb@currsize` `\def\glb@settings{%`

When `\glb@settings` gains control a size change was requested and all previous font assignments need to be replaced. Therefore the old values of the fonts are no longer needed. For every *math group* the new assignments are appended to `\math@fonts`. But this happens only if the `math@fonts` switch is set to true. However, we always set up the correct math sizes for script and scriptscript fonts since they may be needed even if we don't set up the whole math machinery.

Here we set the math size, script size and scriptscript size. If the `S@...` macro is not defined we have to first calculate the three sizes.

```
218 \expandafter\ifx\csname S@\f@size\endcsname\relax
219 \calculate@math@sizes
220 \fi
```

The effect of this is that `\calculate@math@sizes` may or may not define the `S@...` macro. In the first case the next time the same size is requested this macro is used, otherwise `\calculate@math@sizes` is called again. This also sets the `math@fonts` switch. If it is true we must switch the math fonts.

```
221 \csname S@\f@size\endcsname
222 \ifmath@fonts
223 {*trace}
224 \ifnum \tracingfonts>\tw@
225 \font@info{Setting up math fonts for
226 \f@size/\f@baselineskip}\fi
227 }
```

Inside a group we execute the macro for the current math *version*. This sets `\math@fonts` to a list of `\textfont...` assignments. `\getanddefine@fonts` (which may be called at this point) needs the `\escapechar` parameter to be set to `-1`.

```
228 \begingroup
229 \escapechar\m@ne
230 \csname mv@\math@version \endcsname
```

Then we set `\globaldefs` to 1 so that all following changes are done globally. The math font assignments recorded in `\math@fonts` are executed and `\glb@currsize` is set equal to `\f@size`. This signals that the fonts for math in this size are set up.

```
231 \globaldefs\@ne
232 \math@fonts
```

```

233 \let \glb@currsize \f@size
234 \endgroup

```

Finally we execute any code that is supposed to happen whenever the math font setup changes. This register will be executed in local mode which means that everything that is supposed to have any effect should be done globally inside. We can't execute it within `\globaldefs\one` as we don't know what ends up inside this register, e.g., it might contain calculations which use some local registers to calculate the final (global) value.

```

235 \the\every@math@size

```

Otherwise we announce that the math fonts are not set up for this size.

```

236 {*trace}
237 \else
238 \ifnum \tracingfonts>\tw@
239 \font@info{No math setup for
240 \f@size/\f@baselineskip}\fi
241
```

```

242 \fi
243 }
244
```

*(End definition for `\glb@settings` and `\glb@currsize`.)*

`\baselinestretch` In `\selectfont` we used `\baselinestretch` as a factor when assigning a value to `\baselineskip`. We use 1 as a default (i.e. no stretch).

```

245
```

```

246 \def\baselinestretch{1}
```

*(End definition for `\baselinestretch`.)*

`\every@math@size` We must still define the hook `\every@math@size` we used in `\glb@settings`. We initialize it to nothing. It is important to remember that everything that goes into this hook should to global updates, local changes will have weird effects.

```

247 \newtoks\every@math@size
248 \every@math@size={}
249
```

*(End definition for `\every@math@size`.)*

## 5.2 Math fonts setup

### 5.2.1 Outline of algorithm for math font sizes

TeX uses the math fonts that are current when the end of a formula is reached. If we don't want to keep font setups local to every formula (which would result in an enormous overhead, we have to be careful not to end up with the wrong setup in case formulas are nested, e.g., we need to be able to handle

```
$ a=b+c \mbox{ \small for all } b \text{ and } c \text{ in } Z }
```

Here the inner formulae `b` and `c\in Z` are typeset in `\small` but we have to return to `\normalsize` before we reach the closing `$` of the outer formula.

This is handled in the following way:

- At any point in the document the global variable `\glb@currsize` contains the point size for which the math fonts currently are set up.

2. Whenever we start a formula we compare its value with the local variable `\f@size` that describes the current text font size.
3. If both are the same we assume that we can use the current math font setup without adjustment.
4. If they differ we call `\gbl@settings` which changes the math font setup and updates `\gbl@currsize`.
  - (a) If we are recursively inside another formula (`\if@inmath`) we ensure that `\gbl@settings` is executed again in the outer formula, so that the old setup is automatically restored.
  - (b) Otherwise, we set the switch `@inmath` locally to `true` so that all nested formulae will be able to detect that they are nested in some outer formula.

The above algorithm has the following features:

- For sizes which are not containing any formula no math setup is done. Compared to the original algorithm of NFSS this results in the following savings:
  - No unnecessary loading of math fonts for sizes that are not used to typeset any math formulae (explicit or implicit ones).
  - No time overhead due to unnecessary changes of the math font setup on entrance and exit of the text font size.
- Math font setup changes for top-level formulae will survive (there is no restoration after the formula) thus any following formula in the same size will be directly typesettable. Compared to original implementation in NFSS2 the new algorithm has the overhead of one test per formula to see if the current math setup is valid (in the original algorithm the setup was always valid, thus no test was necessary).
- In nested formulae the math font setup is restored in the outer formula by a series of `\aftergroup` commands and checks. Compared to the original algorithm this involves additional checks ( $2 \times \langle \text{non-math levels} \rangle$  per inner formula).

### 5.2.2 Code for math font size setting

`\check@mathfonts` In the `\check@mathfonts` macros we implement the steps 2 to 4 except that instead of a switch the macro `\init@restore@glb@settings` is used.

```

250 {*2ekernel | package}
251 \def\check@mathfonts{%
252 \ifx \gbl@currsize \f@size
253 {*trace}
254 \ifnum \tracingfonts>\thr@@
255 \o@font@info{*** MATH: no change \f@size\space
256 curr/global (\curr@math@size/\gbl@currsize)}\fi
257 {*}trace}
258 \else
259 {*trace}
260 \ifnum \tracingfonts>\thr@@
261 \o@font@info{*** MATH: setting up \f@size\space
262 curr/global (\curr@math@size/\gbl@currsize)}\fi
263 {*}trace}

```

```

264 \glb@settings
265 \init@restore@glb@settings
266 \fi
267 \let\curr@math@size\f@size
268 \def\init@restore@glb@settings{\aftergroup\restglb@settings}%
269 }

(End definition for \check@mathfonts.)
```

`\init@restore@glb@settings` This macros does by default nothing but get redefined inside `\check@mathfonts` to initiate fontsize restoring in nested formulas.

```

270 <-trace> \let\init@restore@glb@settings\relax
271 <*trace>
272 \def\init@restore@glb@settings{%
273 \ifnum \tracingfonts>\thr@@
274 \o@font@info{*** MATH: no resetting (not in
275 nested math)}\fi
276 }
277 </trace>
```

(End definition for `\init@restore@glb@settings`.)

`\restglb@settings` This macro will be executed the first time after the current formula.

```

278 \def\restglb@settings{%
279 <*trace>
280 \ifnum \tracingfonts>\thr@@
281 \o@font@info{*** MATH: restoring}\fi
282 </trace>
283 \begingroup
284 \let\f@size\curr@math@size
285 \ifx\glb@currsize\f@size
286 <*trace>
287 \ifnum \tracingfonts>\thr@@
288 \o@font@info{*** MATH: ... already okay (\f@size)}\fi
289 </trace>
290 \else
291 <*trace>
292 \ifnum \tracingfonts>\thr@@
293 \o@font@info{*** MATH: ... to \f@size}\fi
294 </trace>
295 \glb@settings
296 \fi
297 \endgroup
298 }
```

(End definition for `\restglb@settings`.)

### 5.2.3 Other code for math

`\use@mathgroup` The `\use@mathgroup` macro should be used in user macros to select a math group. Depending on whether or not the `margid` option is in force it has two or three arguments. For this reason it should be called as the last macro.

First we test if we are inside math mode since we don't want to apply a useless definition.

```
299 \def\use@mathgroup#1#2{\relax\ifmmode
```

```

300 {*trace}
301 \ifnum \tracingfonts>\tw@
302 \count@#2\relax
303 \@font@info{Using \noexpand\mathgroup
304 (\the\count@) #2}\fi
305
```

If so we first call the ‘=’ macro (i.e. argument three) to set up special things for the selected math group. Then we call `\mathgroup` to select the group given by argument two and finally we place #1 (i.e. the argument of the *math alphabet identifier*) at the end. This part of the code is surrounded by two commands which behave like `\begingroup` and `\endgroup` if we want *math alphabet identifier*s but will expand into `\empty` if we want simply switches to a new math group. Since argument number 2 may be a digit instead of a control sequence we add a `\relax`. Otherwise something like `\mit{1}` would switch to math group 11 (and back) instead of printing an oldstyle 1.

```

306 \math@bgroup
307 \expandafter\ifx\csname M@\f@encoding\endcsname#1\else
308 #1\fi
309 \mathgroup#2\relax

```

Before we reinsert the swallowed token (arg. three) into the input stream, in the case that the *math alphabet identifier* isn’t called in math mode, we remove the `\fi` with the `\expandafter` trick. This is necessary if the token is actually an macro with arguments. In such a case the `\fi` will be misinterpreted as the first argument which would be disastrous.

```
310 \expandafter\math@egroup\fi}%

```

The surrounding macros equal `\begingroup` and `\endgroup`. But using internal names makes it possible to overwrite their meaning in certain cases. This is for example used in *AMS-T<sub>E</sub>X* macros for placing accents.

(End definition for `\use@mathgroup`.)

`\math@egroup` If the `margid` option is in force (which can be tested by looking at the definition of `\math@bgroup` we change the `\math@egroup` command a bit to display the current *math group number* after it closes the scope of *math alphabet* with `\endgroup`.

```

311 {*trace}
312 \ifx\math@bgroup\bgroup
313 \def\math@egroup#1{\egroup
314 \ifnum \tracingfonts>\tw@
315 \@font@info{Restoring \noexpand\mathgroup
316 (\ifnum\mathgroup=\m@ne default\else \the\mathgroup \fi)%
317 }\fi}
318 \fi
319
```

(End definition for `\math@egroup`.)

`\getanddefine@fonts` `\getanddefine@fonts` has two arguments: the *math group number* and the *family/series/shape* name as a control sequence.

```
320 \def\getanddefine@fonts#1#2{%

```

First we turn of tracing when `\tracingfonts` is less than 4.

```

321 <+debug> \pushtracing
322 <+debug> \ifnum\tracingfonts<4 \tracingoff
323 <+debug> \else \tracingon\getanddefine@fonts \fi

```

```

324 {*trace}
325 \ifnum \tracingfonts>\tw@
326 \count@#1\relax
327 \@font@info{\noexpand\mathgroup (\the\count@) #1 :=\MessageBreak
328 \string#2 \tf@size/\sf@size/\ssf@size}\fi
329
```

We append the current `\tf@size` to #2 to obtain the font name.<sup>27</sup> Again, `font@name` is defined globally, for the reasons explained in the description of `\wrong@fontshape`.

```

330 \xdef\font@name{\csname \string#2/\tf@size\endcsname}%

```

Then we call `\pickup@font` to load it if necessary. We remember the internal name as `\textfont@name`.

```

331 \pickup@font \let\textfont@name\font@name

```

Same game for `\scriptfont` and `\scriptscriptfont`:

```

332 \xdef\font@name{\csname \string#2/\sf@size\endcsname}%
333 \pickup@font \let\scriptfont@name\font@name
334 \xdef\font@name{\csname \string#2/\ssf@size\endcsname}%
335 \pickup@font

```

Then we append the new `\textfont...` assignments to the `\math@fonts`.

```

336 \edef\math@fonts{\math@fonts
337 \textfont#1\textfont@name
338 \scriptfont#1\scriptfont@name
339 \scriptscriptfont#1\font@name}%

```

Just before ending this macro we have to pop the tracing stack if it was pushed before.

```

340 (+debug) \poptracing
341 }
342
```

*(End definition for `\getanddefine@fonts`.)*

## 6 Scaled font extraction

`\ifnot@nil` We begin with a simple auxiliary macro. It checks whether its argument is the token `\@nil`. If so, it expands to `\gobble` which discards the following argument, otherwise it expands to `\firstofone` which reproduces its argument.

```

343 {*2ekernel}
344 \def\ifnot@nil#1{\def\reserved@a{#1}%
345 \ifx\reserved@a\@nil \expandafter\gobble
346 \else \expandafter\firstofone\fi}

```

*(End definition for `\ifnot@nil`.)*

`\remove@to@nnil` Three other auxiliary macros will be needed in the following: `\remove@to@nnil` gobbles up everything up to, and including, the next `\@nnil` token, and `\remove@angles` and `\remove@star` do the same for the character `>` and `*`, respectively, instead of `\@nnil`.

```

347 \def\remove@to@nnil#1\@nnil{#1}
348 \def\remove@angles#1>{\set@simple@size@args{#1}}
349 \def\remove@star#1*{#1}

```

---

<sup>27</sup>One might ask why this expansion does not generate a macro name that starts with an additional `\` character. The solution is that `\escapechar` is set to `-1` before `\getanddefine@fonts` is called.

(End definition for `\remove@to@nnil`, `\remove@angles`, and `\remove@star`.)

- `\extract@sizefn` This macro takes a size specification and parses it into size function and the optional and mandatory arguments.

```
350 \def\extract@sizefn#1#2\@nil{%
351 \if#2>\set@size@funct@args#1\@nil
352 \let\sizefn@info\empty
353 \else\expandafter\set@size@funct@args\remove@star#2\@nil
354 \def\sizefn@info{#1}\fi
355 }
```

(End definition for `\extract@sizefn`.)

- `\try@simple@size` This function tries to extract the given size (specified by `\f@size`) for the requested font shape. The font information must already be present in `\font@info`. The central macro that does the real work is `\extract@fontinfo`. We will first give a simple example how this macro works, and describe it in full generality later.

Assume that the requested parameters are: *encoding scheme* ‘OT1’, *family* ‘cm’, *series* ‘sansserif’, *shape* ‘normal’, and *size* ‘12’. The corresponding font definitions have already been extracted from the macro `\OT1/cm/sansserif/normal` and stored in `font@info`. (Otherwise `\extract@fontinfo` doesn’t get called.) This information consists of a token list made of characters of category code 12 of the form

```
<10*>cmss10<12*>cmss12<17*>cmss17
```

For reasonable packages one usually needs more sizes but this is sufficient to get the flavour. We will define a macro `\extract@fontinfo` to find the external font name (‘cmss12’) for us:

```
\def\extract@fontinfo#1<12*#2>#3<#4\@nnil{%
 \set@simple@size@args#3<#4\@nnil
 \execute@size@function{#2}}
```

so that when it gets called via

```
\extract@fontinfo<10*>cmss10<12*>cmss12<17*>cmss17\@nnil
```

#1 will contain all characters before `<12*>`, #2 will be empty, #3 will be exactly `cmss12`, and #3 will be `17>cmss17`. The expansion is therefore

```
\set@simple@size@args cmss12<17*>cmss17\@nnil
\execute@size@function{}
```

This means: the default (empty) size function will be executed, with its optional argument set to empty and its mandatory argument set to `cmss12` by `\set@simple@size@args`. As we discussed earlier, the effect of the default size function is to load the given external font (`cmss12`) at the specified size (12)—which is exactly what was intended.

But this is only part of the whole story. It may be that the size requested does not occur in the token list `\font@info`. And the simple definition of `\extract@fontinfo` we gave above does not allow to specify give more than one size specification in front of the external font name.

Let’s address these two problems separately. The first one is solved with the following trick: We define `\extract@fontinfo` as follows:

```
\def\extract@fontinfo#1<#2>#3<#4\@nnil{%
 \ifnot@nil{#3}%
 {\set@simple@size@args#3<#4\@nnil
 \execute@size@function{#2}%
 }%
}
```

How does this work? We call `\extract@fontinfo` via

```
\expandafter\extract@fontinfo\font@info<12*>\@nil<\@nnil
```

i.e. by appending `<12*>\@nil<\@nnil`. If the size ('12' in this case) appears in `\font@info` everything works as explained above, the only difference being that argument #4 of `\extract@fontinfo` additionally gets the tokens `<12*>\@nil<\@nnil`. However, if the size is not found everything up to the final `<12*>` is in argument #1, #3 gets `\@nil`, and #2 and #4 are empty. The macro `\ifnot@nil` will discard the calls to `\set@simple@size@args` and `\execute@size@function`, and hence `\font@info` will continue to be equal to `\@empty`. This means that no simple size specification matching the requested size could be found.

The second problem (more than one simple size specification for one external font name) will be addressed in `\set@simple@size@args` below.

The macros are hidden inside other control sequences so that we have to build `\extract@fontinfo` in several steps.

So here's the actual definition of `\extract@font` in `\try@simple@size`.

```
356 % % this could be replaced by \try@size@range making the subst slower!
357 \def\try@simple@size{%
```

`\reserved@a` is made an abbreviation for the head of the definition of the macro `\extract@fontinfo`.

```
358 \def\reserved@a{\def\extract@fontinfo####1}{%
```

Now we can define `\extract@fontinfo`. Here we handle a small but convenient variation: in case of the default (empty) size function it is allowed to omit the \* character.

```
359 \expandafter\reserved@a\expandafter<\f@size>##2<##3\@nnil{%
360 \ifnot@nil{##2}%
361 {\set@simple@size@args##2<##3\@nnil
362 \execute@size@function\sizefn@info
363 }%
}
```

Now we call `\extract@fontinfo`. Note the `<\@nil` tokens at the end.

```
364 \expandafter\expandafter
365 \expandafter\extract@fontinfo\expandafter\font@info
366 \expandafter<\f@size>\@nil<\@nnil
367 }
```

(End definition for `\try@simple@size`.)

`\set@simple@size@args` As promised above, the macro `\set@simple@size@args` will handle the case of several size specifications in a row. If another size specification follows, the very first token of its argument list is the character <. By starting the definition as follows,

```
368 \def\set@simple@size@args#1<%
```

parameter #1 is empty in this case, and contains the size function's arguments otherwise. We distinguish these two cases (Note that the character < cannot appear in #1) by calling `\remove@angles` for empty #1 and `\extract@sizefn` otherwise. In the latter case we have to take care of the remaining character tokens and discard them. This is done by `\remove@to@nnil`. Note also the use of Kabelschacht's method.

```

369 \if<#1<%
370 \expandafter\remove@angles
371 \else
372 \extract@sizefn#1*\@nil
373 \expandafter\remove@to@nnil
374 \fi}

```

*(End definition for `\set@simple@size@args`.)*

Now, we are through with the case of a simple size, except for calling the size function. This will be handled later, as it is the same mechanism for all types of size specification. We will now proceed to macros for extraction of size range specification.

`\extract@rangefontinfo` `\extract@rangefontinfo` goes through a font shape definition in the input until it recognizes the tokens <`\@nil`>. It looks for font ranges with font size functions. Its operation is rather simple: it discards everything up to the next size specification and passes this on to `\is@range` for inspection. The specification (parameter #2 is inserted again, in case it is needed later.

```

375 \def\extract@rangefontinfo#1<#2>{%
376 \is@range#2->\@nil#2}

```

*(End definition for `\extract@rangefontinfo`.)*

`\is@range` `\is@range` is again a sort of dispatcher macro: if the size specification it is looking at is not a range specification it discards it and calls `\extract@rangefontinfo` to continue the search. Otherwise it calls `\check@range` to check the requested size against the specified range.

From the way `\is@range` is called inside `\extract@rangefontinfo` we see that #2 is the character > if the size specification found is a simple one (as it does not contain a - character. This is checked easily enough and `\extract@rangefontinfo` called again. Note that the extra tokens inserted after the `\@nil` in the call to `\is@range` appear at the beginning of the first argument to `\extract@rangefontinfo` and are hence ignored.

```

377 \def\is@range#1-#2\@nil{%
378 \if>#2\expandafter\check@singl\else
379 \expandafter\check@range\fi}

```

*(End definition for `\is@range`.)*

`\check@range` `\check@range` takes lower bound as parameter #1, upper bound as #2, size function as #3 and the size function's arguments as #4. If #3 is the special token `\@nil` `\font@info` is exhausted and we can stop searching.

```

380 \def\check@range#1-#2>#3<#4\@nnil{%
381 \ifnot@nil{#3}{%

```

If #3 wasn't `\@nil` we have a range. We start by assuming that we have to recurse. Note that we have to reinsert an < as it was already removed by scanning.

```

382 \def\reserved@ff{\extract@rangefontinfo<#4\@nnil}%

```

We have to make sure that both boundaries are present, if not we have to set them. Here we check the upper bound. If `\upper@bound` is zero after the assignment we set it to `\maxdimen` (upper open range). We need to use a  $\langle dimen \rangle$  register for the scan since we may have a decimal number as the boundary.

```
383 \upper@bound0#2\p@
384 \ifdim\upper@bound=\z@ \upper@bound\maxdimen\fi
```

Now we check the upper boundary against `\f@size`. If it is larger or equal than `\f@size` this range is no good and we have to recurse.

```
385 \ifdim \f@size \p@<\upper@bound
```

Otherwise we have to check the lower bound. This time it is not necessary to scan the boundary value into a register because if it is empty we get zero as desired. We could even omit the 0 which would result in 1pt as default lower boundary. If `\f@size` is smaller than the boundary we have to recurse.

```
386 \lower@bound0#1\p@
387 \ifdim \f@size \p@<\lower@bound
388 \else
```

If both tests are passed we can try executing the size function.

```
389 \set@simple@size@args#3<#4\@nnil
390 \execute@size@function\sizefn@info
```

If the function was successful it should have left an external font name in `\external@font`. We use this to see if we can stop scanning. Otherwise we recurse.

```
391 \ifx\external@font\@empty
392 \else
393 \let\reserved@f\@empty
394 \fi
395 \fi
396 \fi
397 \reserved@f}}
```

(End definition for `\check@range`.)

`\lower@bound` We use two dimen registers `\lower@bound` and `\upper@bound` to store the lower and upper endpoints of the range we found.

```
398 \newdimen\lower@bound
399 \newdimen\upper@bound
```

(End definition for `\lower@bound` and `\upper@bound`.)

`\check@single` `\check@single` takes the size as parameter #1, size function as #2 and the size function's arguments as #3. We can assume that there is always something in the pipeline since the very last entry is a faked range (see above).

```
400 \def\check@single#1>#2<#3\@nnil{%
```

We start by assuming that we have to recurse. Note that we have to reinsert an < as it was already removed by scanning.

```
401 \def\reserved@f{\extract@fontinfo<#3\@nnil}%
```

Now we check the size against `\f@size`. If it is not equal `\f@size` it is no good and we have to recurse.

```
402 \ifdim \f@size \p@=#1\p@
```

Otherwise if this test is passed we can try executing the size function.

```
403 \set@size@args#2<#3\@nil
404 \execute@size@function\sizefn@info
```

If the function was successful it should have left an external font name in `\external@font`. We use this to see if we can stop scanning. Otherwise we recurse.

```
405 \ifx\external@font\@empty
406 \else
407 \let\reserved@f\@empty
408 \fi
409 \fi
410 \reserved@f}
```

(End definition for `\check@single`.)

`\set@size@funct@args` This macro sets the optional and mandatory arguments for a size function. If the optional argument is not present it is set to the empty token list. The mandatory argument is delimited by the token `\@nil`.

```
411 \def\set@size@funct@args{\@ifnextchar[%]
412 \set@size@funct@args{\set@size@funct@args[]}}
413 \def\set@size@funct@args[#1]{\@nil{%
414 \def\mandatory@arg{#1}%
415 \def\optional@arg{}%
416 }}
```

(End definition for `\set@size@funct@args` and `\set@size@funct@args@`.)

`\DeclareSizeFunction` This function defines a new size function hiding the internal from the designer. The body of the size function may use `\optional@arg` and `\mandatory@arg` denoting the optional and mandatory argument that may follow the size specification `<...>`.

```
417 {*2ekernel}
418 \def\DeclareSizeFunction#1#2{\@namedef{s@fct@#1}{#2}}
419 \@onlypreamble\DeclareSizeFunction
420
```

(End definition for `\DeclareSizeFunction`.)

`\execute@size@function` This macro is very simple. The only point worth noting is that calling an undefined size function will do nothing (actually execute a `\relax`).

```
421 {*2ekernel | package}
422 \def\execute@size@function#1{%
423 {*trace}
424 \@ifundefined{s@fct@#1}{%
425 {\errmessage{Undefined font size function #1}%
426 \s@fct@}%
427 {\csname s@fct@#1\endcsname}%
428 }%
429 {-trace} \csname s@fct@#1\endcsname
430 }
431
```

(End definition for `\execute@size@function`.)

\try@size@range This macro tries to find a suitable range for requested size (specified by \f@size) in \font@info. All the relevant action is done in \extract@rangefontinfo. All that needs to be done is to stuff in the token list in \font@info so that \extract@rangefontinfo can inspect it. Note the <-\*\@nil> token at the end to stop scanning.

```

432 {*2ekernel}
433 \def\try@size@range{%
434 \expandafter\extract@rangefontinfo\font@info <-*>\@nil<\@nnil
435 }

```

(End definition for \try@size@range.)

\try@size@substitution This is the last thing that can be tried. If the desired \f@size is found neither among the simple size specifications nor in one of the ranges the whole list of size specifications is searched for a nearby simple size.

```
436 \gdef\try@size@substitution{%
```

First we do some initializations. \tempdimb will hold the difference between the wanted size and the best solution found so far, so we initialise it with \maxdimen. The macro \best@size will hold the best size found, nothing found is indicated by the empty value.

```

437 \tempdimb \maxdimen
438 \let \best@size \empty

```

Now we loop over the specification

```

439 \expandafter \try@simples \font@info <\number\@M\@nil<\@nnil
440 }

```

(End definition for \try@size@substitution.)

\font@submax \fontsubfuzz The macro \font@submax records the maximal deviation from the desired size encountered so far. Its value is used in a warning message at \end{document}. The macro \fontsubfuzz contains the amount that will not cause terminal warnings (warnings still go into the transcript file).

```

441 \def\font@submax{0pt}
442 \def\fontsubfuzz{.4pt}
443 {/2ekernel}
444 {+package}\def\fontsubfuzz{0pt}

```

(End definition for \font@submax and \fontsubfuzz.)

\try@simples \try@simples goes through a font shape definition in the input until it recognizes the tokens <\*\@nil>. It looks for simple sizes to determine the two closest sizes. It is assumed that simple sizes are in increasing order.

```

445 {*2ekernel}
446 \gdef\try@simples#1<#2>{%
447 \tryif@simple#2->\tryif@simple}

```

(End definition for \try@simples.)

\tryis@simple \tryis@simple is similar to \is@range. If it sees a simple size, it checks it against the value of \f@size and sets \lower@font@size or \higher@font@size. In the latter case, it stops the iteration. By adding <\number\@M> at the end of the line we always have an end point. This is a hack which probably should be corrected.

First it checks whether it is finished already, then whether the size specification in question is a simple one.

```
448 \gdef\tryif@simple#1-#2\tryif@simple{%
```

Most common case for `\reserved@f` first:

```
449 \let \reserved@f \try@simples
450 \if>#2%
```

If so, it compares it to the value of `\f@size`. This is done using a dimen register since there may be fractional numbers.

```
451 \dimen@ #1\p@
452 \ifdim \dimen@<\OM\p@
```

If `\dimen@` is `\OM\p@` we have reached the end of the fontspec (hopefully) otherwise we compare the value with `\f@size` and compute in `\@tempdimc` the absolute value of the difference between the two values.

```
453 \ifdim \f@size\p@<\dimen@
454 \@tempdimc \dimen@
455 \advance\@tempdimc -\f@size\p@
456 \else
457 \@tempdimc \f@size\p@
458 \advance\@tempdimc -\dimen@
459 \fi
```

The result is then compared with the smallest difference we have encountered, if the new value (in `\@tempdimc` is smaller) we have found a size which is a better approximation so we make it the `\best@size` and adjust `\@tempdimb`.

```
460 \ifdim \@tempdimc<\@tempdimb
461 \@tempdimb \@tempdimc
462 \def \best@size{#1}%
463 \fi
```

When we have reached the end of the fontspec we substitute the best size found (if any). We code this inline to save macro space; in the past this was done by a macro called `\subst@size`.

```
464 \else
```

This macro substitutes the size recorded in `\best@size` for the unavailable size `\f@size`. `\font@submax` records the maximum difference between desired size and selected size in the whole run.

```
465 % %\subst@size %% coded inline
466 % %\def\subst@size{%
467 \ifx \external@font\empty
468 \ifx \best@size\empty
469 \else
470 \ifdim \@tempdimb>\font@submax \relax
471 \xdef \font@submax {\the\@tempdimb}%
472 \fi
473 \let \f@user@size \f@size
474 \let \f@size \best@size
475 \ifdim \@tempdimb>\fontsubfuzz\relax
476 \font@warning{Font\space shape\space
477 '\curr@fontshape'\space in\space size\space
478 <\f@user@size>\space not\space available\MessageBreak
479 size\space <\f@size>\space substituted}%
480 \fi
481 \try@simple@size
482 \do@subst@correction
```

```

483 \fi
484 \fi
485 % %}

```

This brings us back into the main part of `\tryif@simple`. Finally we get rid of any rubbish left over on the input stack.

```

486 \let \reserved@f \remove@to@nnil
487 \fi
488 \fi

```

If it's a range iterate also.

```
489 \reserved@f}
```

(End definition for `\tryis@simple` and `\subst@size`.)

## 6.1 Sizefunctions

In the following we define some useful size functions.

- `\s@fct@` This is the default size function. Mandatory argument is an external font name, optional argument a scale factor. The font is scaled to `\f@size` if no optional argument is present, and to `\f@size` multiplied by the optional argument otherwise.

```

490 \DeclareSizeFunction{}{\empty@sfcnt\@font@warning}
491 \DeclareSizeFunction{s}{\empty@sfcnt\@font@info}

492 \def\empty@sfcnt#1{%
493 \tempdimb \f@size\p@
494 \ifx\optional@arg\empty
495 \else
496 \tempdimb \optional@arg\tempdimb
497 #1{Font}space shape\space '\curr@fontshape'\space
498 will\space be\MessageBreak
499 scaled\space to\space size\space \the\tempdimb}%
500 \fi
501 \edef\external@font{\mandatory@arg\space at\the\tempdimb}}

```

(End definition for `\s@fct@`.)

- `\s@fct@gen` This size function generates the external name from the mandatory argument and the requested user size, and thus can be used for external names where the size is encoded in the font name. The optional argument a scale factor. The font is scaled to `\f@size` if no optional argument is present, and to `\f@size` multiplied by the optional argument otherwise.

```

502 \DeclareSizeFunction{gen}{\gen@sfcnt\@font@warning}
503 \DeclareSizeFunction{sgen}{\gen@sfcnt\@font@info}

504 \def\gen@sfcnt{%
505 \edef\mandatory@arg{\mandatory@arg\f@size}%
506 \empty@sfcnt}

```

(End definition for `\s@fct@gen` and `\s@fct@sgen`.)

\s@fct@genb This size function is similar to `gen`, but for fonts where the size is encoded in the font name in centipoins, as in the DC fonts version 1.2. The font is scaled to `\f@size` if no optional argument is present, and to `\f@size` multiplied by the optional argument otherwise.

```

507 \DeclareSizeFunction{genb}{\genb@sfcnt\@font@warning}
508 \DeclareSizeFunction{sgenb}{\genb@sfcnt\@font@info}
509 \def\genb@sfcnt{%
510 \edef\mandatory@arg{\mandatory@arg\expandafter\genb@x\f@size..\@0}%
511 \empty@sfcnt}

```

(End definition for `\s@fct@genb` and `\s@fct@sgenb`.)

\genb@x The auxiliary macros `\genb@x` and `\genb@y` are used to convert the `\f@size` into centi-points.

```

512 \def\genb@x#1.#2.#3{\two@digits{#1}\genb@y#200\@0}
513 \def\genb@y#1#2#3{\@0{#1#2}}

```

(End definition for `\genb@x` and `\genb@y`.)

\s@fct@sub This size function handles font substitution. The mandatory argument is a family/series/shape combination, the optional argument (if present) is ignored. The font encoding scheme cannot be changed. Therefore, the first thing we do is to prepend the encoding scheme.

```

514 \DeclareSizeFunction{sub}{\sub@sfcnt\@font@warning}
515 \DeclareSizeFunction{ssub}{\sub@sfcnt\@font@info}
516 \def\sub@sfcnt#1{%
517 \edef\mandatory@arg{\f@encoding\mandatory@arg}%

```

Next action is split the arg into its individual components and allow for a late font shape load.

```

518 \begingroup
519 \expandafter\split@name\mandatory@arg/\@nil
520 \try@load@fontshape
521 \endgroup

```

Then we record the current `\f@size` since it may get clobbered.

```
522 \let\f@user@size\f@size
```

Then we check whether this new combination is defined and give an error message if not. In this case we also switch to `\error@fontshape`.

```

523 \expandafter
524 \ifx\csname\mandatory@arg\endcsname\relax
525 \errmessage{No\space declaration\space for\space
526 shape\space \mandatory@arg}%
527 \error@fontshape
528 \else

```

Otherwise we warn the user about the substitution taking place.

```

529 #1{Font\space shape\space '\curr@fontshape'\space in\space
530 size\space <\f@size>\space not\space available\MessageBreak
531 Font\space shape\space '\mandatory@arg'\space tried\space
532 instead}%
533 \expandafter\split@name\mandatory@arg/\@nil
534 \fi

```

Then we restart the font specification scan by calling `\get@external@font`.

```
535 \edef\f@size{\f@user@size}%
536 \get@external@font
```

Finally `\do@subst@correction` is called to get the font name right.

```
537 \do@subst@correction
538 }
```

(*End definition for \s@fct@sub.*)

- `\@font@aliasinfo` Sometimes a substitution is only done to map a long font name to a standard shape or series, e.g.,

```
DeclareFontShape{T1}{Roboto-LF}{b}{it}{<-> alias * Roboto-LF/bold/it}{}
```

Using the `ssub` function in that case will give a strange (and incorrect) warning. As an alternative we therefore offer the size function `alias`. It will still add some info into the `.log` file, but no longer complains that the font shape is not available. It is implemented by grabbing the default warning text and replacing it with a new one.

```
539 </2ekernel>
540 <*2ekernel | latexrelease>
541 <latexrelease>\IncludeInRelease{2020/02/02}%
542 <latexrelease> {\@font@aliasinfo}{alias size function}%
543 \DeclareSizeFunction{alias}{\sub@sfcnt@\font@aliasinfo}
544 \def@\font@aliasinfo#1{%
545 \@font@info{Font\space shape\space ‘\curr@fontshape’\space
546 aliased\space to\MessageBreak ‘\mandatory@arg’}%
547 }
548 </2ekernel | latexrelease>
549 <latexrelease>\EndIncludeInRelease
550 <latexrelease>\IncludeInRelease{0000/00/00}%
551 <latexrelease> {\@font@aliasinfo}{alias size function}%
552 <latexrelease>\let\s@fct@alias@\undefined
553 <latexrelease>\let@\font@aliasinfo@\undefined
554 <latexrelease>
555 <latexrelease>\EndIncludeInRelease
556 <*2ekernel>
```

(*End definition for \@font@aliasinfo.*)

- `\s@fct@subf` The `subf` size function allows substitution of another font. The mandatory argument is the external name of the font to be substituted, the optional argument a size scaling factor like in the default size function. The main difference to the default size function is the warning message.

```
557 \DeclareSizeFunction{subf}{\sub@sfcnt@\font@warning}
558 \DeclareSizeFunction{ssubf}{\subf@sfcnt@\font@info}

559 \def\subf@sfcnt#1{%
560 #1{Font\space shape\space ‘\curr@fontshape’\space in\space
561 size\space f@size\space not\space available\MessageBreak
562 external\space font\space ‘\mandatory@arg’\space used}%
563 \empty@sfcnt#1%
564 }
```

(*End definition for \s@fct@subf.*)

- \s@fct@fixed The **fixed** size function is for using a font at a different size than requested. A warning message is printed, and the external font to be used is taken from the mandatory argument. If an optional argument is present it is used as the ‘at’ size for the font. Otherwise the font is loaded at its design size.

```

565 \DeclareSizeFunction{fixed}{\fixed@sfcnt\@font@warning}
566 \DeclareSizeFunction{sfixed}{\fixed@sfcnt\@font@info}
567 \def\fixed@sfcnt#1{%
568 \ifx\optional@arg\empty
569 \let\external@font\mandatory@arg
570 \else
571 \edef\external@font{\mandatory@arg\space at\optional@arg pt}%
572 \fi
573 #1{External\space font\space '\external@font'\space loaded\space
574 for\space size\MessageBreak
575 <\f@size>}%
576 }
577
```

(End definition for \s@fct@fixed.)

## File y

# ltfsscmp.dtx

This file contains the implementation of commands giving compatibility with the original ‘NFSS1’ release of the Font Selection Scheme.

**Warning:** The macro documentation is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

Version 1 of NFSS is obsolete now for about 20 years (and was “current” only for a short intermediate time) so with the 2015 release these internal interface commands are removed from the kernel and made available via `latexrelease` package so that backward compatibility remains ensured for very old documents.

```
1 {*latexrelease}
2 \IncludeInRelease{2015/01/01}{\new@fontshape}%
3 {NFSS version1 commands}%
4 \let\new@fontshape@undefined
5 \let\warn@rel@i@undefined
6 \let\scan@fontshape@undefined
7 \let\scan@@fontshape@undefined
8 \let\subst@fontshape@undefined
9 \let\extra@def@undefined
10 \let\default@mextra@undefined
11 \let\preload@sizes@undefined
12 \let\err@rel@i@undefined
13 \let\newmathalphabet@undefined
14 \let\newmathalphabet@undefined
15 \let\newmathalphabet@@@@undefined
16 \let\if@no@font@opt@undefined
17 \let\@no@font@opt@false@undefined
18 \let\define@mathalphabet@undefined
19 \let\define@mathgroup@undefined
20 \let\addtoversion@undefined
21 \EndIncludeInRelease
```

In older releases we provide the original definitions.

```
22 \IncludeInRelease{0000/00/00}{\new@fontshape}%
23 {NFSS version1 commands}%
```

`\new@fontshape` The interface is now `\DeclareFontShape`.

```
24 \gdef\new@fontshape#1#2#3#4{%
25 \warn@rel@i\new@fontshape\DeclareFontShape
26 \expandafter\scan@fontshape\@gobble#4<\@nil><<%
27 \DeclareFontShape U{#1}{#2}{#3}\reserved@f}%
28 \onlypreamble\new@fontshape
```

(*End definition for `\new@fontshape`.*)

`\warn@rel@i` The warning message used above.

```
29 \gdef\warn@rel@i#1#2{%
30 \font@warning{*** NFSS release 1 command}
```

```

31 \noexpand#1found\MessageBreak
32 *** Update by using release 2 command
33 \string#2.\MessageBreak
34 *** Recovery is probably possible}%
35 }%
36 \onlypreamble\warn@rel@i

```

(End definition for \warn@rel@i.)

\scan@fontshape This will scan the old font shape definition syntax.

```

37 \gdef\scan@fontshape{%
38 \let\reserved@f\empty
39 \let\reserved@e\empty % holds last info
40 \scan@@fontshape
41 }%
42 \onlypreamble\scan@fontshape

```

(End definition for \scan@fontshape.)

\scan@@fontshape

```

43 \gdef\scan@@fontshape#1>#2#3<%
44 \ifx\@nil#1%
45 \edef\reserved@f{\reserved@f\reserved@e}%
46 \else
47 \def\reserved@b{#1}% nick names
48 \def\reserved@c{#3}%
49 \in@{ at}{#3}%
50 \ifin@
51 \in@{pt}{#3}%
52 \ifin@ not a proof but a good chance

```

We grab also everything after pt and discard it if people have forgotten to place a percent sign there.

```

53 \def\reserved@a##1 at##2pt##3\@nil{%
54 \def\reserved@b{##2}%
55 \def\reserved@c{##1}%
56 }%
57 \reserved@a#3\@nil
58 \fi
59 \fi
60 \ifnum 0<0#2
61 \edef\reserved@d{subf*\reserved@c}%
62 \ifcase #2\or
63 \or
64 \else
65 \errmessage{*** What's this? NFSS release 0? ***}%
66 \fi
67 \else
68 \edef\reserved@d{#2\reserved@c}%
69 \fi
70 \ifx\reserved@d\reserved@e
71 \edef\reserved@f{\reserved@f<\reserved@b>}%
72 \else
73 \edef\reserved@f{\reserved@f\reserved@e<\reserved@b>}%add old info
74 \let\reserved@e\reserved@d

```

```

75 \fi
76 \expandafter\scan@@fontshape
77 \fi
78 }%
79 \onlypreamble\scan@@fontshape

```

(End definition for `\scan@@fontshape`.)

`\subst@fontshape` This is now also handled by the extend syntax of `\DeclareFontShape`.

```

80 \gdef\subst@fontshape#1#2#3#4#5#6{%
81 \warn@rel@i\subst@fontshape\DeclareFontShape
82 \DeclareFontShape{U}{#1}{#2}{#3}{<->sub*#4/#5/#6}{()}%
83 \onlypreamble\subst@fontshape

```

(End definition for `\subst@fontshape`.)

`\extra@def` This was replaced by `\DeclareFontFamily`.

```

84 \gdef\extra@def#1#2#3{%
85 \warn@rel@i\extra@def\DeclareFontFamily
86 \DeclareFontFamily{U}{#1}{()}%
87 }%
88 \onlypreamble\extra@def

```

(End definition for `\extra@def`.)

`\default@mextra` The new name is `\DeclareFontEncodingDefaults` but in this case we don't feel comfortable with this either.

```

89 \gdef\default@mextra{%
90 \warn@rel@i\default@mextra\DeclareFontEncodingDefaults

```

We pick up the argument to `\default@mextra` implicitly as the second argument of `\DeclareFontEncodingDefaults`.

```

91 \DeclareFontEncodingDefaults\relax
92 }%
93 \onlypreamble\default@mextra

```

(End definition for `\default@mextra`.)

`\preload@sizes` The new interface is `\DeclarePreloadSizes`.

```

94 \gdef\preload@sizes{%
95 \warn@rel@i\preload@sizes\DeclarePreloadSizes
96 \DeclarePreloadSizes U%
97 }%
98 \onlypreamble\preload@sizes

```

(End definition for `\preload@sizes`.)

`\err@rel@i` This macro is used in cases where emulation with NFSS2 features is not really possible.

```

99 \gdef\err@rel@i#1#2{%
100 \o@late@error{*** NFSS release 1 command \noexpand#1found%
101 ^^J*** Recovery not possible. Use \string#2}%
102 {The new release of NFSS doesn't support the
103 \noexpand#1command^^Jany longer.
104 Please upgrade your file to the syntax of NFSS
105 release 2^^Jusing the \noexpand#2command.}%

```

Let's die.

```
106 \batchmode\input.\relax
107 }%
108 \onlypreamble\err@rel@i
```

(End definition for `\err@rel@i`.)

`\newmathalphabet` `\newmathalphabet` is the old form.  
`\newmathalphabet@@` `\gdef\newmathalphabet{%`  
`\newmathalphabet@CC` `\if@no@font@opt`  
110     `\@latex@error{*** NFSS release 1 command`  
111         `\noexpand\newmathalphabet found%`  
112         `^^J \space*** Automatic recovery not possible.%`  
113         `^^J \space*** TYPE H for Help%`  
114         `}%`  
115         `{Please look at the file usrguide.tex for hints on`  
116         `how to resolve this problem.}%`  
117     `\else`  
118         `\warn@rel@i\newmathalphabet\DeclareMathAlphabet`  
119     `\fi`  
120     `\@ifstar\newmathalphabet@CC`  
121         `\newmathalphabet@CC}%`  
122     `\gdef\newmathalphabet@#1{\DeclareMathAlphabet#1{U}{\{}{\}}}%`  
123     `\gdef\newmathalphabet@#1#2#3#4{%`  
124         `\DeclareMathAlphabet{#1}{U}{#2}{#3}{#4}}%`  
125     `\onlypreamble\newmathalphabet`  
126     `\onlypreamble\newmathalphabet@C`  
127     `\onlypreamble\newmathalphabet@CC`

(End definition for `\newmathalphabet`, `\newmathalphabet@@`, and `\newmathalphabet@CC`.)

`\if@no@font@opt`  
`\@no@font@optfalse` `\global\let\if@no@font@opt\iftrue`  
129     `\gdef\@no@font@optfalse{\let\if@no@font@opt\iffalse}%`  
130

(End definition for `\if@no@font@opt` and `\@no@font@optfalse`.)

`\define@mathalphabet` This is a case where dying is best.

```
131 \gdef\define@mathalphabet{%
132 \err@rel@i\define@mathalphabet\DeclareMathAlphabet
133 }%
134 \onlypreamble\define@mathalphabet
```

(End definition for `\define@mathalphabet`.)

`\define@mathgroup` And here is another one

```
135 \gdef\define@mathgroup{%
136 \err@rel@i\define@mathgroup\DeclareSymbolFont
137 }%
138 \onlypreamble\define@mathgroup
```

(End definition for `\define@mathgroup`.)

```
\addtoversion \addtoversion is the old form.
139 \def\addtoversion#1#2{
140 \warn@rel@i\addtoversion\SetMathAlphabet
141 \SetMathAlphabet#2{#1}{U}}%
142 \onlypreamble\addtoversion

(End definition for \addtoversion.)
Finishing off this huge \IncludeInRelease argument:
143 \EndIncludeInRelease
144 </latexrelease>
```

# File z

## ltfssdcl.dtx

This file contains the main implementation of the font selection scheme commands. See other parts of the L<sup>A</sup>T<sub>E</sub>X distribution, or *The L<sup>A</sup>T<sub>E</sub>X Companion* for higher level documentation of these commands.

**Warning:** The macro documentation is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

### 1 Interface Commands

\in@ \in is a utility macro with two arguments. It determines whether its first argument \ifin@ occurs in its second and sets the switch \ifin@ accordingly. The first argument may not contain braces nor # (more precisely, tokens of category code 1, 2, or 6).

```
1 {*2ekernel}
2 \def\in@#1#2%
3 {%
4 \begingroup
5 \def\in@@##1#1{%
6 \toks@\expandafter{\in@@#2{}{}#1}%
7 \edef\in@{\the\toks@}%
8 \expandafter\endgroup
9 \ifx\in@{\emptyset}
10 \in@false
11 \else
12 \in@true
13 \fi
14 }
15 \newif\ifin@
```

(End definition for \in@ and \ifin@.)

Before the \begin{document} command several *math versions* and *math alphabet identifiers* may be declared. In principle, there should be exactly one family/series/shape combination be declared for each version/alphabet pair. But we want to allow for defaults as well for automagical filling of holes.

While building the tables for math alphabet identifiers and math versions we keep several lists:

- the list of all math versions, \version@list, each entry prefixed by the control sequence \version@elt, i.e. this list has the following form

$$\text{\version@elt}\langle\text{version}_1\rangle\text{\version@elt}\langle\text{version}_2\rangle\dots\text{\version@elt}\langle\text{version}_n\rangle$$

- the list of all math alphabet identifiers. Here every entry has the form:

$$\text{\group@elt}\langle\text{math group number}\rangle\\ \{\{\langle\text{default family}\rangle\}\{\langle\text{default series}\rangle\}\{\langle\text{default shape}\rangle\}\}.$$

- Each defined math alphabet identifier holds a list containing Information about the *versions* for which it is defined. This list has a more complicated structure: it looks as follows:

```
\set@alpha<the alphabet identifier itself>
\reserved@c<math version>
...
\@nil
```

where  $\langle font\ info\rangle$  is either  $\reserved@e$  (if the combination is not defined yet) or  
 $\{\{family\}\}\{\{series\}\}\{\{shape\}\}$

`\version@list` We initialize the version list to be empty.

```
16 \let\version@list=\@empty
17 \onlypreamble\version@list
```

(End definition for `\version@list`.)

`\version@elt`

```
18 \let\version@elt\relax
19 \onlypreamble\version@elt
```

(End definition for `\version@elt`.)

`\new@mathversion` The macro `\new@mathversion` is called with the version control sequence as its argument.

```
20 \%def\new@mathversion#1{%
```

The first thing this macro does is to check if the version identifier is already present in `\version@list`. We enclose `\version@list` in braces since it might be empty (if no *version* is defined yet). But this means that we need a suitable number of `\expandafter` primitives.

```
21 \% \expandafter\in@\expandafter#1\expandafter{\version@list}%
22 \% \ifin@
```

If so it prints an error message. The `\next` macro is used to get rid of the four characters `\mv@` that would otherwise appear at the begin of the version name in the error message.

```
23 \% \@latex@error{Math version
24 \% '\expandafter@gobblefour\string#1'
25 \% already defined}\@eha
```

Otherwise we have a new version, and we can proceed with entering it into the tables. We add it to `\version@list`. This is very easy: we define `\version@elt` (which is the delimiter in `\version@list`) to protect itself and the following token from being expanded and simply redefine `\version@list`.

```
26 \% \else
27 \% \global\expandafter\newcount\csname c@\expandafter
28 \% \gobble\string#1\endcsname
29 \% \global\csname c@\expandafter
30 \% \gobble\string#1\endcsname\@ne
31 \% \def\version@elt{\noexpand\version@elt\noexpand}%
32 \% \edef\version@list{\version@list\version@elt#1}%
```

Then we prepare to enter the new version into all math alphabet identifier lists. Remember that these lists use `\reserved@c` as delimiter, and that there appears the control sequence `\reserved@e` that must not be expanded. Therefore we take suitable precautions.

```
33 % \def\reserved@c{\noexpand\reserved@c\noexpand}%
34 % \let\reserved@e\relax
```

We will now go through the `\alpha@list` to process every *(math alphabet identifier)* in turn. Since this list has `\group@elt` as a delimiter we define this control sequence. It has three arguments as every entry consists of three items (as explained above).

```
35 % \def\group@elt##1##2##3{%
```

The first of these arguments is the *(math alphabet identifier)*. We redefine it by appending the information about the new version at the end of the list contained in it. However, there is one subtlety: the definitions for `\reserved@c` and `\reserved@e` made above prevent the main part of the list from being expanded. But we still have to take care of the header and the trailer. To do this we remove the trailer by means of the macro `\remove@nil` which also protect the header from being expanded. Its definition is given below. Now we can prepare to add the new version.

```
36 % \edef##1{\expandafter\remove@nil##1%
37 % \reserved@c
38 % #1%
39 % \reserved@e
40 % \noexpand\@nil}}%
```

Finally we call `\alpha@list` which will now execute the macro `\group@elt` once for every defined *(math alphabet identifier)*. And that's all for now.

```
41 % \alpha@list
42 % \fi}
```

*(End definition for \new@mathversion.)*

`\alpha@list` As we explained above every entry in `\alpha@list` has the form

`\alpha@elt  
(alphabet identifier)<internal group number><default font assignments>...`

We initialize it to `\empty`.

```
43 \let\alpha@list\empty
44 \onlypreamble\alpha@list
```

*(End definition for \alpha@list.)*

`\alpha@elt`

```
45 \let\alpha@elt\relax
46 \onlypreamble\alpha@elt
```

*(End definition for \alpha@elt.)*

`\newgroup` Start the group (fam) allocation at 0. (Doesn't belong here.)

```
47 \count18=-1
```

*(End definition for \newgroup.)*

`\stepcounter`

*(End definition for \stepcounter.)*

\select@group We surround \select@group with braces so that functions using it can be used directly after \_ or ^. However, if we use oldstyle syntax where the math alphabet doesn't have arguments (ie if \math@bgroup is not \bgroup) we need to get rid of the extra group.

```

48 </2ekernel>
49 <latexrelease>\IncludeInRelease{2015/01/01}
50 <latexrelease> {\select@group}{\select@group}%
51 <2ekernel | latexrelease>
52 \def\select@group#1#2#3#4{%
53 \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
54 {%
55 \ifmmode
56 \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
57 \begingroup
58 \escapechar\m@ne
59 \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
60 \globaldefs\@ne \math@fonts
61 \endgroup
62 \init@restore@version
63 \xdef#1{\noexpand\use@mathgroup\noexpand#2%
64 {\number\csname c@mv@\math@version\endcsname}}%
65 \global\advance\csname c@mv@\math@version\endcsname\@ne
66 \else
67 \let#1\relax
68 \@latex@error{Too many math alphabets used in
69 version \math@version}%
70 \@eha
71 \fi
72 \else \expandafter\@non@alpherr\fi
73 #1{#4}%
74 }%
75 }
76 </2ekernel | latexrelease>
77 <latexrelease>\EndIncludeInRelease
78 <latexrelease>\IncludeInRelease{0000/00/00}
79 <latexrelease> {\select@group}{\select@group}%
80 <latexrelease>\def\select@group#1#2#3#4{%
81 <latexrelease> \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
82 <latexrelease> {%
83 <latexrelease> \ifmmode
84 <latexrelease> \ifnum\csname c@mv@\math@version\endcsname<\sixt@n
85 <latexrelease> \begingroup
86 <latexrelease> \escapechar\m@ne
87 <latexrelease> \getanddefine@fonts
88 <latexrelease> {\csname c@mv@\math@version\endcsname}#3%
89 <latexrelease> \globaldefs\@ne \math@fonts
90 <latexrelease> \endgroup
91 <latexrelease> \init@restore@version
92 <latexrelease> \xdef#1{\noexpand\use@mathgroup\noexpand#2%
93 <latexrelease> {\number\csname c@mv@\math@version\endcsname}}%
94 <latexrelease> \global\advance\csname c@mv@\math@version\endcsname\@ne
95 <latexrelease> \else
96 <latexrelease> \let#1\relax
97 <latexrelease> \@latex@error{Too many math alphabets used in
98 version \math@version}%

```

```

99 \langle latexrelease\rangle \@eha
100 \langle latexrelease\rangle \fi
101 \langle latexrelease\rangle \else \expandafter\non@alpherr\fi
102 \langle latexrelease\rangle #1{\#4}%
103 \langle latexrelease\rangle }%
104 \langle latexrelease\rangle
105 \langle latexrelease\rangle\EndIncludeInRelease
106 {*2ekernel}
107 \onlypreamble\restore@mathversion

```

(*End definition for \select@group.*)

\init@restore@version

```

108 \def\init@restore@version{%
109 \global\let\init@restore@version\relax
110 \xdef\restore@mathversion
111 {\expandafter\noexpand\csname mv@\math@version\endcsname
112 \global\csname c@mv@\math@version\endcsname
113 \number\csname c@mv@\math@version\endcsname\relax}%
114 \aftergroup\dorestore@version
115 }
116 \onlypreamble\init@restore@version

```

(*End definition for \init@restore@version.*)

\non@alpherr

```

117 \gdef\non@alpherr#1{\@latex@error{%

```

The command here will have a space at the end of its name, so we make sure not to insert an extra one.

```

118 \string#1allowed only in math mode}\@ehd}

```

(*End definition for \non@alpherr.*)

\dorestore@version

```

119 \def\dorestore@version
120 { \ifmmode
121 \aftergroup\dorestore@version
122 \else
123 \gdef\init@restore@version{%
124 \global\let\init@restore@version\relax
125 \xdef\restore@mathversion
126 {\expandafter\noexpand\csname mv@\math@version\endcsname
127 \global\csname c@mv@\math@version\endcsname
128 \number\csname c@mv@\math@version\endcsname\relax}%
129 \aftergroup\dorestore@version
130 }%
131 \begingroup
132 \let\getanddefine@fonts\@gobbletwo
133 \restore@mathversion
134 \endgroup
135 \fi}%
136 \onlypreamble\dorestore@version

```

(*End definition for \dorestore@version.*)

`\c@localalphabets` To avoid hitting the “no more math fams available” limit of 16, we keep a defined number of math alphabets flexible/local. If we have to allocate any of those we roll back the allocation after the formula has ended, so the next formula can use other alphabets in the slot(s). This makes the processing a bit slower if you are working at the limit, but that is better than dying with “out of memory”.

```

137 </2ekernel>
138 <|latexrelease>\IncludeInRelease{2021/11/15}
139 <|latexrelease> {\document@select@group}{\document@select@group}%
140 {*2ekernel | latexrelease}

```

We don’t really undo the declaration on rollback (as that would be hard to maintain), so rolling forward needs to check if the declaration was already made.

```
141 \ifx\c@localmathalphabets\undefined
```

There is no need to have this counter as part of the include checkpoints, given that it makes little sense to alter its settings mid document. All we want is the ability to change it using the `\setcounter` interface.

By default we keep two math fams flexible.

```

142 \newcount\c@localmathalphabets
143 \setcounter{localmathalphabets}{2}
144 \fi

```

(*End definition for \c@localalphabets.*)

`\document@select@group` The `\document@select@group` command is the version of `\select@group` (inside math versions) that is used in the document body to set up math alphabets (if used).

```

145 \def\document@select@group#1#2#3#4{%
146 \ifx\math@bgroup\math@bgroup\else\relax\expandafter\@firstofone\fi
147 {%
148 \ifmmode

```

We first check if there is still room for allocating another mathgroup. If there is, we check if it can be globally allocated or if we have reached the limit which is given by `\e@mathgroup@top` with `\c@localmathalphabets` subtracted.

```

149 \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
150 \ifnum \numexpr\mathgroup@top-\c@localmathalphabets
151 >\csname c@mv@\math@version\endcsname
152 \else

```

If we are past this point we freeze the current state of the math version so that we can return to it after the formula has ended. Of course, that should be done only once, so we check if `\mv@<version>@frozen` already exists.

```
153 \ifcsname mv@\math@version @frozen\endcsname \else
```

We have to pass the current value of `\math@version` not the macro itself, because some of the processing is delayed to a point where the value may have changed again—not doing this caused a puzzling error in one setup.

```

154 \expandafter\freeze@math@version\expandafter{\math@version}%
155 \fi
156 \fi
157 \begingroup
158 \escapechar\m@ne
159 \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
160 \globaldefs\one \math@fonts

```

```

161 \endgroup
162 \expandafter\extract@alph@from@version
163 \csname mv@\math@version\expandafter\endcsname
164 \expandafter{\number\csname
165 c@mv@\math@version\endcsname}%
166 #1%
167 \global\advance\csname c@mv@\math@version\endcsname\@ne
168 \else
169 \let#1\relax
170 \@latex@error{Too many math alphabets used in
171 version \math@version}%
172 \@eha
173 \fi

```

Extra `\expandafter` to remove the `\expandafter` added below

```
174 \else \expandafter\expandafter\expandafter\non@alpherr\fi
```

We surround `\select@group` with braces so that functions using it can be used directly after `_` or `^`.

If the legacy interface is used, e.g., `$_\sf -1$` the math alphabet `#1` does not take an argument so we better do not surround `#4` with braces, because then we get `\relax` into the formula and introduce an extra Ord atom. The two different cases can be distinguished by looking at the current value of `\math@bgroup`.

```

175 \expandafter#1\ifx\math@bgroup\bgroup{#4}\else#4\fi
176 }%
177 }

```

This command stores the current state of the math version and sets things up to return to it after each formula from now on. We use L3 programming layer code to set it up.

```

178 \ExplSyntaxOn
179 \cs_new_protected:Npn\freeze@math@version #1 {

```

Save the current `\mv@<version>` code and the number of allocated mathgroups inside.

```

180 @font@info{Freeze~ math~ alphabet~ allocation~ in~ version-
181 #1.\MessageBreak
182 Allocated-math-groups:~\int_use:c{ c@mv@ #1 }~%
183 (local:~ \int_use:N\c@localmathalphabets)~}
184 \cs_gset_eq:cc { mv@#1@frozen }{ mv@#1 }
185 \tl_gset:cx { g__nfss_frozen_mv_ #1 _tl }{ \int_use:c { c@mv@#1 } }

```

Doing the reset the first time, we wait until we are out of math mode, so we use some recursive `\group_insert_after:N` for this before we execute `\mv@<version>@reset`.

```

186 \group_insert_after:N __nfss_init_mv_freeze:N \exp_after:wN
187 \group_insert_after:N \cs:w mv@#1@reset \cs_end:

```

The `\check@mathfonts` is called at the very beginning of each math formula, so it is a good way to hook in the resetting. Again that has to happen after the formula has ended, but we know because of the place where `\check@mathfonts` is used that a single `\aftergroup` is sufficient.

```

188 \tl_gput_right:No \check@mathfonts
189 {
190 \exp_after:wN \group_insert_after:N \cs:w mv@#1@reset \cs_end:
191 }

```

Here is the definition of `\mv@<version>@reset`. If there has been no new math alphabet allocation, doing a reset would just cause a lot of unnecessary processing, so we do a quick check upfront for this.

```

192 \cs_gset:cpn{\mv@#1@reset}
193 {
194 \int_compare:nNnTF { \int_use:c{c@mv@#1} } >
195 { \tl_use:c{g__nfss_frozen_mv_ #1 _tl} }
196 {
197 @font@info{Undo~math~alphabet~allocation~in~version~#1}
198 }
199
```

If the undo is necessary, we restore the `\mv@<version>` code.

```

200 \cs_gset_eq:cc { mv@#1 }{ mv@#1@frozen }
201 \int_gset:cn { c@mv@#1 }{ \tl_use:c {g__nfss_frozen_mv_ #1 _tl} }
202
```

But we also should undo changes to the math alphabet definitions. We therefore run this code with a modified definition for `\getanddefine@fonts` because there is no need to do anything to the symbol fonts that are permanently allocated.

```

203 \group_begin:
204 \cs_set_eq:NN \getanddefine@fonts \use_none:nn
205 \use:c {mv@#1}
206 \group_end:
207
```

If there was no change, we report that in the log (but this branch could go completely).

```

208 @font@info{No~math~alphabet~change~to~frozen~version~#1}
209 }
210
```

To do the initial freeze in a safe place, we check if we are in math mode and if so try again after the group has ended by pushing the command and its single token argument with two `\group_insert_after:N`s after the current group. If we are no longer in math mode we bypass the conditional and so the next token is our argument which is then finally executed.

```

211 \cs_new_protected:Npn __nfss_init_mv_freeze:N #1 {%
212 \mode_if_math:T { \group_insert_after:N __nfss_init_mv_freeze:N
213 \group_insert_after:N } #1
214 }

215 \ExplSyntaxOff
216 </2ekernel | latexrelease>
217 <latexrelease>\EndIncludeInRelease
218 <latexrelease>\IncludeInRelease{2020/10/01}
219 <latexrelease>\{\document@select@group\}\{\document@select@group\}%
220 <latexrelease>\def\document@select@group#1#2#3#4{%
221 <latexrelease>\ifx\math@bgroup\bgroupl\else\relax\expandafter\@firstofone\f
222 <latexrelease>\{%
223 <latexrelease>\ifmmode
224 <latexrelease>\ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
225 <latexrelease>\begingroup
226 <latexrelease>\escapechar\m@ne
227 <latexrelease>\getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
```

```

228 <|latexrelease> \globaldefs\@ne \math@fonts
229 <|latexrelease> \endgroup
230 <|latexrelease> \expandafter\extract@alph@from@version
231 <|latexrelease> \csname mv@\math@version\expandafter\endcsname
232 <|latexrelease> \expandafter{\number\csname
233 <|latexrelease> c@mv@\math@version\endcsname}%
234 <|latexrelease> #1%
235 <|latexrelease> \global\advance\csname c@mv@\math@version\endcsname\@ne
236 <|latexrelease> \else
237 <|latexrelease> \let#1\relax
238 <|latexrelease> \@latex@error{Too many math alphabets used
239 <|latexrelease> in version \math@version}%
240 <|latexrelease> \@eha
241 <|latexrelease> \fi
242 <|latexrelease> \else \expandafter\expandafter\expandafter\non@alpherr\fi
243 <|latexrelease> \expandafter#1\ifx\math@bgroup\bgroup{#4}\else#4\fi
244 <|latexrelease> }%
245 <|latexrelease> }
246 <|latexrelease>\EndIncludeInRelease
247 <|latexrelease>\IncludeInRelease{2015/01/01}
248 <|latexrelease> {\document@select@group}{\document@select@group}%
249 <|latexrelease>
250 <|latexrelease>\def\document@select@group#1#2#3#4{%
251 <|latexrelease> \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
252 <|latexrelease> {%
253 <|latexrelease> \ifmmode
254 <|latexrelease> \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
255 <|latexrelease> \begingroup
256 <|latexrelease> \escapechar\m@ne
257 <|latexrelease> \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
258 <|latexrelease> \globaldefs\@ne \math@fonts
259 <|latexrelease> \endgroup
260 <|latexrelease> \expandafter\extract@alph@from@version
261 <|latexrelease> \csname mv@\math@version\expandafter\endcsname
262 <|latexrelease> \expandafter{\number\csname
263 <|latexrelease> c@mv@\math@version\endcsname}%
264 <|latexrelease> #1%
265 <|latexrelease> \global\advance\csname c@mv@\math@version\endcsname\@ne
266 <|latexrelease> \else
267 <|latexrelease> \let#1\relax
268 <|latexrelease> \@latex@error{Too many math alphabets used
269 <|latexrelease> in version \math@version}%
270 <|latexrelease> \@eha
271 <|latexrelease> \fi
272 <|latexrelease> \else \non@alpherr\fi
273 <|latexrelease> #1{#4}%
274 <|latexrelease> }%
275 <|latexrelease> }
276 <|latexrelease>\EndIncludeInRelease
277 <|latexrelease>
278 <|latexrelease>\IncludeInRelease{0000/00/00}
279 <|latexrelease> {\document@select@group}{\document@select@group}%
280 <|latexrelease>
281 <|latexrelease>\def\document@select@group#1#2#3#4{%

```

```

282 〈\latexrelease〉 \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
283 〈\latexrelease〉 {%
284 〈\latexrelease〉 \ifmmode
285 〈\latexrelease〉 \ifnum\csname c@mv@math@version\endcsname<\sixt@on
286 〈\latexrelease〉 \begingroup
287 〈\latexrelease〉 \escapechar\m@ne
288 〈\latexrelease〉 \getanddefine@fonts
289 〈\latexrelease〉 {\csname c@mv@math@version\endcsname}#3%
290 〈\latexrelease〉 \globaldefs\@ne \math@fonts
291 〈\latexrelease〉 \endgroup
292 〈\latexrelease〉 \expandafter\extract@alph@from@version
293 〈\latexrelease〉 \csname mv@math@version\expandafter\endcsname
294 〈\latexrelease〉 \expandafter{\number\csname
295 〈\latexrelease〉 c@mv@math@version\endcsname}%
296 〈\latexrelease〉 #1%
297 〈\latexrelease〉 \global\advance\csname c@mv@math@version\endcsname\@ne
298 〈\latexrelease〉 \else
299 〈\latexrelease〉 \let#1\relax
300 〈\latexrelease〉 \@latex@error{Too many math alphabets used
301 〈\latexrelease〉 in version \math@version}%
302 〈\latexrelease〉 \Oeha
303 〈\latexrelease〉 \fi
304 〈\latexrelease〉 \else \expandafter\non@alpherr\fi
305 〈\latexrelease〉 #1{#4}%
306 〈\latexrelease〉 }%
307 〈\latexrelease〉 }
308 〈\latexrelease〉 \EndIncludeInRelease
309 〈*2ekernel〉

(End definition for \document@select@group, \freeze@math@version, and
__nfss_init_mv_freeze:N.)

```

### \process@table

```

310 \def\process@table{%
311 \def\cdp@elt##1##2##3##4{%
312 \o@font@info{Checking defaults for
313 ##1##2##3##4}%
314 \expandafter
315 \ifx\csname##1##2##3##4\endcsname\relax

```

Grouping is important for two reasons, first \cdp@elt will get redefined if \Declare... functions are executed within the external .fd file and secondly \try@load@fontshape changes a lot of catcodes without surrounding itself with a group.

```

316 \begingroup
317 \def\f@encoding{##1}\def\f@family{##2}%
318 \try@load@fontshape
319 \endgroup
320 \fi
321 \expandafter
322 \ifx\csname##1##2##3##4\endcsname\relax
323 \@latex@error{This NFSS system isn't set up properly}%
324 {For encoding scheme ##1 the defaults
325 ##2##3##4 do not form a valid font shape}%
326 \else
327 \o@font@info{... okay}%

```

```

328 \fi}%
329 \cdp@list

```

Now we make sure that `\error@fontshape` is okay.

```

330 \begingroup
331 \escapechar\m@ne
332 \error@fontshape
333 \expandafter\ifx\csname \curr@fontshape\endcsname\relax
334 \begingroup
335 \try@load@fontshape
336 \endgroup
337 \fi
338 \expandafter\ifx\csname \curr@fontshape\endcsname\relax
339 \@latex@error{This NFSS system isn't set up properly}%
340 {The system maintainer forgot to specify a suitable
341 substitution
342 font shape using the \noexpand\DeclareErrorFont
343 command}%
344 \fi
345 \endgroup

```

Set `\select@group` to its meaning used within the document body.

```

346 \let\select@group\document@select@group

```

Install the default font attributes as they are currently pointing to error font face. We can speed up the process by just using `\edef`, thereby avoiding all kind of extra processing. Don't use `\reset@font` since that would trigger `\selectfont`.

```

347 \fontencoding\encodingdefault
348 \edef\f@family{\familydefault}%
349 \edef\f@series{\seriesdefault}%
350 \edef\f@shape{\shapedefault}%

```

Drop stuff not longer needed. We need to add many more!!!!!!

```

351 \everyjob{}%
352 }
353 \onlypreamble\process@table

```

*(End definition for `\process@table`.)*

```

354 %\onlypreamble\set@mathradical

```

`\DeclareMathVersion`

```

355 </2ekernel>
356 <*2ekernel | latexrelease>
357 <| latexrelease>\IncludeInRelease{2021/11/15}%
358 <| latexrelease> {\DeclareMathVersion}{local alphabets}%
359 \def\DeclareMathVersion#1{%

```

When declaring a new math version we need to instantiate an L3 variable that is used when we freeze the version, because too many alphabets got allocated. If we don't do this, L3 programming layer complains if it is run in checking mode.

```

360 \namedef{g__nfss_frozen_mv_#1_tl}{}%
361 \expandafter\new@mathversion\csname mv@#1\endcsname}
362 \onlypreamble\DeclareMathVersion
363 </2ekernel | latexrelease>
364 <| latexrelease>\EndIncludeInRelease

```

```

365 \langle latexrelease \rangle \IncludeInRelease{0000/00/00}%
366 \langle latexrelease \rangle \def\DeclareMathVersion{\local alphabets}%
367 \langle latexrelease \rangle \def\DeclareMathVersion#1{%
368 \langle latexrelease \rangle \expandafter\new@mathversion\csname mv@#1\endcsname}
369 \langle latexrelease \rangle \EndIncludeInRelease
370 \langle *2ekernel \rangle

```

(End definition for \DeclareMathVersion.)

\new@mathversion

```

371 \def\new@mathversion#1{%
372 \expandafter\in@\expandafter#1\expandafter{\version@list}%
373 \ifin@
374 \font@info{Redeclaring math version
375 \expandafter\gobblefour\string#1'}%
376 \else
377 \expandafter\newcount\csname c@\expandafter
378 \gobble\string#1\endcsname
379 \def\version@elt{\noexpand\version@elt\noexpand}%
380 \edef\version@list{\version@list\version@elt#1}%
381 \fi

```

\toks@ is used to gather all tokens for the math version. \count@ will be used to count the math groups we add to this version.

```

382 \toks@{}%
383 \count@0z@%

```

Now we loop over \group@list to add all math groups defined so far to the version and at the same time to count them.

```

384 \def\group@elt##1##2{%
385 \advance\count@\@ne
386 \addto@hook\toks@{\getanddefine@fonts##1##2}%
387 }%
388 \group@list

```

We set the counter for this math version to the number of math groups found in \group@list.

```
389 \global\csname c@\expandafter\gobble\string#1\endcsname\count@%
```

Now we loop over \alpha@list to add all math alphabets known so far. We have to distinguish the case that an alphabet by default should produce an error in new versions.

```

390 \def\alpha@elt##1##2##3{%
391 \ifx##2\no@alphabet@error
392 \toks@\expandafter{\the\toks@\install@mathalphabet##1%
393 \no@alphabet@error##1}%
394 \else
395 \toks@\expandafter{\the\toks@\install@mathalphabet##1%
396 \select@group##1##2##3}%
397 \fi
398 }%
399 \alpha@list

```

Finally we define the math version to expand to the contents of \toks@.

```

400 \xdef#1{\the\toks@}%
401 }
402 \onlypreamble\new@mathversion

```

(End definition for `\new@mathversion`.)

`\DeclareSymbolFont`

```
403 \def\DeclareSymbolFont#1#2#3#4#5{%
404 \tempswafalse
405 \edef\reserved@b{#2}%
406 \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
407 \ifx\reserved@b\reserved@c \tempswatrue\fi}%
408 \cdp@list
409 \if@tempswa
410 \ifundefined{sym#1}{%
411 \ifnum\count18<15 %
412 \expandafter\new@mathgroup\csname sym#1\endcsname
413 \expandafter\new@symbolfont\csname sym#1\endcsname
414 {#2}{#3}{#4}{#5}%
415 \else
416 \@latex@error{Too many symbol fonts declared}\@eha
417 \fi
418 }%
419 }%
420 \font@info{Redeclaring symbol font '#1'}%
```

Update the group list.

```
421 \def\group@elt##1##2{%
422 \noexpand\group@elt\noexpand##1%
423 \expandafter\ifx\csname sym#1\endcsname##1%
424 \expandafter\noexpand\csname#2/#3/#4/#5\endcsname
425 \else
426 \noexpand##2%
427 \fi}%
428 \xdef\group@list{\group@list}%
```

Update the version list.

```
429 \def\version@elt##1{%
430 \expandafter
431 \SetSymbolFont@\expandafter##1\csname#2/#3/#4/#5\expandafter
432 \endcsname \csname sym#1\endcsname
433 }%
434 \version@list
435 }%
436 \else
437 \@latex@error{Encoding scheme '#2' unknown}\@eha
438 \fi
439 }
440 \onlypreamble\DeclareSymbolFont
```

(End definition for `\DeclareSymbolFont`.)

`\group@list`

```
441 \let\group@list\empty
442 \onlypreamble\group@list
```

(End definition for `\group@list`.)

```

\group@elt
443 \let\group@elt\relax
444 \@onlypreamble\group@elt

(End definition for \group@elt.)
```

```

\new@symbolfont
445 \def\new@symbolfont#1#2#3#4#5{%
446 \toks@\expandafter{\group@list}%
447 \edef\group@list{\the\toks@\noexpand\group@elt\noexpand#1%
448 \expandafter\noexpand\csname#2/#3/#4/#5\endcsname}%
449 \def\version@elt##1{\toks@\expandafter{##1}%
450 \edef##1{\the\toks@\noexpand\getanddefine@fonts
451 #1\expandafter\noexpand\csname#2/#3/#4/#5\endcsname}%
452 \global\advance\csname c@\expandafter
453 \gobble\string##1\endcsname\@ne
454 }%
455 \version@list
456 }
457 \@onlypreamble\new@symbolfont

(End definition for \new@symbolfont.)
```

```

\SetSymbolFont
458 \def\SetSymbolFont#1#2#3#4#5#6{%
459 \tempswafalse
460 \edef\reserved@b{#3}%
461 \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
462 \ifx\reserved@b\reserved@c \tempswatrue\fi}%
463 \cdp@list
464 \if@tempswa
465 \expandafter\SetSymbolFont@%
466 \csname mv@#2\expandafter\endcsname\csname#3/#4/#5/#6\expandafter
467 \endcsname \csname sym#1\endcsname
468 \else
469 \@latex@error{Encoding scheme '#3' unknown}\@eha
470 \fi
471 }
472 \@onlypreamble\SetSymbolFont

(End definition for \SetSymbolFont.)
```

```

\SetSymbolFont@
473 \def\SetSymbolFont@#1#2#3{%
474 \expandafter\in@\expandafter#1\expandafter{\version@list}%
475 \ifin@
476 \expandafter\in@\expandafter#3\expandafter{\group@list}%
477 \ifin@
478 \begingroup
479 \expandafter\get@cdp\string#2\@nil\reserved@a
480 \toks@{}%
481 \def\install@mathalphabet##1##2{%
482 \addto@hook\toks@{\install@mathalphabet##1##2}%
483 }%

```

```

484 \def\getanddefine@fonts##1##2{%
485 \ifnum##1=#3%
486 \addto@hook\toks@{\getanddefine@fonts#3#2}%
487 \expandafter\get@cdp\string##2\@nil\reserved@b
488 \ifx\reserved@a\reserved@b\else
489 \@font@info{Encoding ‘\reserved@b’ has changed
490 to ‘\reserved@a’ for symbol font\MessageBreak
491 ‘\expandafter\gobblefour\string#3’ in the
492 math version ‘\expandafter
493 \@gobblefour\string#1’}%
494 \fi
495 \@font@info{%
496 Overwriting symbol font
497 ‘\expandafter\gobblefour\string#3’ in
498 version ‘\expandafter
499 \@gobblefour\string#1’}\MessageBreak
500 \@spaces \expandafter\@gobble\string##2 -->
501 \expandafter\@gobble\string##2}%
502 \else
503 \addto@hook\toks@{\getanddefine@fonts##1##2}%
504 \fi}%
505 #1%
506 \xdef#1{\the\toks@}%
507 \endgroup
508 \else
509 \@latex@error{Symbol font ‘\expandafter\gobblefour\string#3’
510 not defined}\@eha
511 \fi
512 }%
513 \@latex@error{Math version ‘\expandafter\gobblefour\string#1’
514 is not
515 defined}{You probably misspelled the name of the math
516 version.^^JOr you have to specify an additional package.}%
517 \fi
518 }
519 \onlypreamble\SetSymbolFont@
```

(End definition for `\SetSymbolFont@`.)

```

\get@cdp
520 \def\get@cdp#1#2/#3\@nil#4{\def#4{#2}}
521 \onlypreamble\get@cdp
```

(End definition for `\get@cdp`.)

`\DeclareMathAlphabet`

```

522 \def\DeclareMathAlphabet#1#2#3#4#5{%
523 \tempswafalse
524 \edef\reserved@b{#2}%
525 \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
526 \ifx\reserved@b\reserved@c \tempswatrue\fi}%
527 \cdp@list
528 \if@tempswa
529 \expandafter\ifx
530 \csname\expandafter\string#1\endcsname
```

```

531 \relax
532 \new@mathalphabet#1{#2}{#3}{#4}{#5}%
533 \else
534 \edef\reserved@a{\noexpand\in@\{\string\select@group\}%
535 {\expandafter\meaning\csname \expandafter
536 \@gobble\string#1\space\endcsname}\}%
537 \reserved@a
538 \ifin@
539 \font@info{Redeclaring math alphabet \string#1}%
540 \def\version@elt##1{%
541 \expandafter\SetMathAlphabet@expandafter
542 ##1\csname#2/#3/#4/#5\expandafter\endcsname
543
544 \csname M@#2\expandafter\endcsname
545 \csname \expandafter\@gobble\string#1\space\endcsname#1}%
546 \version@list
547 \else

```

Check if it is already a math alphabet.

```

547 \edef\reserved@a{\noexpand\in@\{\string\use@mathgroup\}%
548 {\expandafter\meaning\csname \expandafter
549 \@gobble\string#1\space\endcsname}\}%
550 \reserved@a
551 \ifin@

```

In that case overwriting is simple since there is nothing inserted in the math version macros.

```

552 \font@info{Redeclaring math alphabet \string#1}%
553 \new@mathalphabet#1{#2}{#3}{#4}{#5}%

```

Otherwise panic.

```

554 \else
555 \@latex@error{Command '\string#1' already defined}\@eha
556 \fi
557 \fi
558 \fi
559 \else
560 \@latex@error{Encoding scheme '#2' unknown}\@eha
561 \fi
562 }
563 \onlypreamble\DeclareMathAlphabet

```

*(End definition for \DeclareMathAlphabet.)*

```

\new@mathalphabet
564 \def\new@mathalphabet#1#2#3#4#5{%
565 \toks@\expandafter{\alpha@list}%
566 \edef#1{\expandafter\noexpand\csname \expandafter
567 \@gobble\string#1\space\endcsname
568 \if/#5/%
569 \noexpand\no@alphabet@error
570 \noexpand\no@alphabet@error
571 \else
572 \expandafter\noexpand\csname M@#2\endcsname

```

```

573 \expandafter\noexpand\csname#2/#3/#4/#5\endcsname
574 \fi
575 }%
576 \toks2\expandafter{#1}%
577 \edef\alpha@list{\the\toks@\noexpand\alpha@elt\the\toks2}%
578 \def\version@elt##1{\toks@\expandafter{##1}%
579 \edef##1{\the\toks@\install@mathalphabet
580 \expandafter\noexpand
581 \csname \expandafter\@gobble
582 \string#1\space\endcsname
583 \if/#5/%
584 \noexpand\no@alphabet@error
585 \noexpand#1%
586 \else
587 \noexpand\select@group\the\toks2
588 \fi} }%
589 }%
590 \version@list
591 \expandafter\edef\csname \expandafter\@gobble
592 \string#1\space\endcsname{\if/#5/%
593 \noexpand\no@alphabet@error
594 \noexpand#1%
595 \else
596 \noexpand\select@group\the\toks2
597 \fi} }%
598 \edef#1{\noexpand\protect
599 \expandafter\noexpand\csname \expandafter
600 \@gobble\string#1\space\endcsname}%
601 }
602 \onlypreamble\new@mathalphabet

```

(End definition for `\new@mathalphabet`.)

### `\SetMathAlphabet`

```

603 \def\SetMathAlphabet#1#2#3#4#5#6{%
604 \tempswafalse
605 \edef\reserved@b{#3}%
606 \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
607 \ifx\reserved@b\reserved@c \tempswatrue\fi}%
608 \cdp@list
609 \if@tempswa
610 \expandafter\SetMathAlphabet@%
611 \csname mv@#2\expandafter\endcsname\csname#3/#4/#5/#6\expandafter
612 \endcsname \csname M@#3\expandafter\endcsname
613 \csname \expandafter\@gobble\string#1\space\endcsname#1%
614 \else
615 \@latex@error{Encoding scheme '#3' unknown}\@eha
616 \fi
617 }
618 \onlypreamble\SetMathAlphabet

```

(End definition for `\SetMathAlphabet`.)

### `\SetMathAlphabet@`

```

619 \def\SetMathAlphabet{\#1\#2\#3\#4\#5{%
620 \expandafter\in@\expandafter#1\expandafter{\version@list}%
621 \ifin@
622 \expandafter\in@\expandafter#4\expandafter{\alpha@list}%
623 \ifin@
624 \begingroup
625 \toks@{}%
626 \def\getanddefine@fonts##1##2{%
627 \addto@hook\toks@{\getanddefine@fonts##1##2}%
628 }%
629 \def\reserved@c##1##2##3##4{%
630 \expandafter\@gobble\string##4}%
631 \def\install@mathalphabet##1##2{%
632 \ifx##1#4%
633 \addto@hook\toks@%
634 {\install@mathalphabet#4{\select@group#4#3#2}}%
635 \font@info{Overwriting math alphabet
636 'string#5' in version '\expandafter
637 \@gobblefour\string#1'\MessageBreak
638 \spaces \reserved@c##2 -->
639 \expandafter\@gobble\string#2}%
640 \else
641 \addto@hook\toks@{\install@mathalphabet##1{##2}}%
642 \fi
643 }%
644 #1%
645 \xdef#1{\the\toks@}%
646 \endgroup
647 \else

```

If the math alphabet was defined via `\DeclareSymbolFontAlphabet` we have remove its external definition and add it as a normal math alphabet to every version before trying to change it in one version.

```

648 \edef\reserved@a{%
649 \noexpand\in@{\string\use@mathgroup}{\meaning#4}%
650 \reserved@a
651 \ifin@
652 \def\reserved@b##1\use@mathgroup##2##3{%
653 \def\reserved@b##3\def\reserved@c##2}%
654 \expandafter\reserved@b#4%
655 \begingroup
656 \def\install@mathalphabet##1##2{%
657 \addto@hook\toks@{\install@mathalphabet##1{##2}}%
658 }%
659 \def\getanddefine@fonts##1##2{%
660 \addto@hook\toks@{\getanddefine@fonts##1##2}%
661 \ifnum##1=\reserved@c
662 \expandafter
663 \addto@hook\expandafter\toks@%
664 \expandafter{\expandafter\install@mathalphabet
665 \expandafter#4\expandafter
666 \expandafter\select@group\expandafter
667 #4\reserved@c}%
668 \fi

```

```

669 }%
670 \def\version@elt##1{%
671 \toks@{}%
672 ##1%
673 \xdef##1{\the\toks@}%
674 }%
675 \version@list
676 \endgroup

```

Put it into the `\alpha@list` with default ‘error’

```

677 \expandafter\gdef\expandafter\alpha@list\expandafter
678 {\alpha@list
679 \alpha@elt #4\no@alphabet@error \no@alphabet@error}%
680 \gdef#4{\no@alphabet@error #5}% fake things :-

```

Then call the internal setting routine again:

```

681 \SetMathAlphabet@{#1}{#2}{#3}#4#5%
682 \else
683 @latex@error{Command ‘\string#5’ not defined as a
684 math alphabet}%
685 {Use \noexpand\DeclareMathAlphabet to define it.}%
686 \fi
687 \fi
688 \else
689 @latex@error{Math version ‘\expandafter\@gobblefour\string#1’
690 is not
691 defined}{You probably misspelled the name of the math
692 version.^~JOr you have to specify an additional package.}%
693 \fi
694 }
695 \onlypreamble\SetMathAlphabet@

```

(End definition for `\SetMathAlphabet@`.)

`\DeclareMathAccent` Could do with more checks like allowing single number in #4 lowercase in #4 etc

```

696 </2ekernel>
697 <*2ekernel | latexrelease>
698 <latexrelease>\IncludeInRelease{2019/10/01}%
699 <latexrelease> {DeclareMathAccent}{Make math accents robust}%
700 \def\DeclareMathAccent#1#2#3#4{%
701 \expandafter\in@\csname sym#3\expandafter\endcsname
702 \expandafter{\group@list}%
703 \ifin@
704 \begingroup
705 \count\z@=#4\relax
706 \count\tw@\count\z@
707 \divide\count\z@\sixt@@n
708 \count@\count\z@
709 \multiply\count@\sixt@@n
710 \advance\count\tw@-\count@
711 \if\relax\noexpand#1% is command?
712 \edef\reserved@a{\noexpand\in@
713 {\expandafter\@gobble\string\mathaccent}
714 {\expandafter\meaning
715 \csname\expandafter\@gobble\string#1\space\endcsname}}%

```

```

716 \reserved@a
717 \ifin@
718 \expandafter\let
719 \csname\expandafter\@gobble\string#1\space\endcsname
720 \undefined
721 \expandafter\set@mathaccent
722 \csname sym#3\endcsname#1#2%
723 {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
724 \font@info{Redeclaring math accent \string#1}%
725 \else
726 \expandafter\ifx
727 \csname\expandafter\@gobble\string#1\endcsname
728 \relax
729 \expandafter\set@mathaccent
730 \csname sym#3\endcsname#1#2%
731 {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
732 \else
733 \@latex@error{Command ‘\string#1’ already defined}\@eha
734 \fi
735 \fi
736 \else
737 \@latex@error{Not a command name: ‘\noexpand#1’}\@eha
738 \fi
739 \endgroup
740 \else
741 \@latex@error{Symbol font ‘#3’ is not defined}\@eha
742 \fi
743 }
744 </2ekernel | latexrelease>
745 <latexrelease>\EndIncludeInRelease
746 <latexrelease>\IncludeInRelease{0000/00/00}%
747 <latexrelease> {DeclareMathAccent}{Make math accents robust}%
748 <latexrelease>\def\DeclareMathAccent#1#2#3#4{%
749 <latexrelease> \expandafter\in@{\csname sym#3\expandafter\endcsname
750 <latexrelease> \expandafter{\group@list}%
751 <latexrelease> \ifin@
752 <latexrelease> \begin{group}
753 <latexrelease> \count\z@=#4\relax
754 <latexrelease> \count\tw@\count\z@
755 <latexrelease> \divide\count\z@\sixt@@n
756 <latexrelease> \count@\count\z@
757 <latexrelease> \multiply\count@\sixt@@n
758 <latexrelease> \advance\count\tw@-\count@
759 <latexrelease> \if\relax\noexpand#1% is command?
760 <latexrelease> \edef\reserved@a{\noexpand\in@
761 <latexrelease> {\expandafter\@gobble\string\mathaccent}{\meaning#1}}%
762 <latexrelease> \reserved@a
763 <latexrelease> \ifin@
764 <latexrelease> \expandafter\set@mathaccent
765 <latexrelease> \csname sym#3\endcsname#1#2%
766 <latexrelease> {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
767 <latexrelease> \font@info{Redeclaring math accent \string#1}%
768 <latexrelease> \else
769 <latexrelease> \expandafter\ifx

```

```

770 <|latexrelease> \csname\expandafter\@gobble\string#1\endcsname
771 <|latexrelease> \relax
772 <|latexrelease> \expandafter\set@mathaccent
773 <|latexrelease> \csname sym#3\endcsname#1#2%
774 <|latexrelease> {\hexnumber@{\count\z@\hexnumber@{\count\tw@}}%}
775 <|latexrelease> \else
776 <|latexrelease> \@latex@error{Command ‘\string#1’ already defined}\@eha
777 <|latexrelease> \fi
778 <|latexrelease> \fi
779 <|latexrelease> \else
780 <|latexrelease> \@latex@error{Not a command name: ‘\noexpand#1’}\@eha
781 <|latexrelease> \fi
782 <|latexrelease> \endgroup
783 <|latexrelease> \else
784 <|latexrelease> \@latex@error{Symbol font ‘#3’ is not defined}\@eha
785 <|latexrelease> \fi
786 <|latexrelease> \}
787 <|latexrelease>\EndIncludeInRelease
788 <|*2ekernel>
789 \onlypreamble\DeclareMathAccent

```

(*End definition for \DeclareMathAccent.*)

\set@mathaccent

```

790 </|2ekernel>
791 <|*2ekernel | latexrelease>
792 <|latexrelease>\IncludeInRelease{2019/10/01}%
793 <|latexrelease> {\set@mathaccent}{makemath accents robust}%
794 \def\set@mathaccent#1#2#3#4{%
795 \xdef#2{\mathaccent"\mathchar@type#3\hexnumber@#1#4\relax}%
796 \MakeRobust#2%
797 }
798 \onlypreamble\set@mathaccent
799 </|2ekernel | latexrelease>
800 <|latexrelease>\EndIncludeInRelease
801 <|latexrelease>\IncludeInRelease{0000/00/00}%
802 <|latexrelease> {\set@mathaccent}{makemath accents robust}%
803 <|latexrelease>
804 <|latexrelease>\def\set@mathaccent#1#2#3#4{%
805 \xdef#2{\mathaccent"\mathchar@type#3\hexnumber@#1#4\relax}%
806 <|latexrelease>
807 <|latexrelease>\EndIncludeInRelease
808 <|*2ekernel>

```

(*End definition for \set@mathaccent.*)

\DeclareMathSymbol

```

809 \def\DeclareMathSymbol#1#2#3#4{%
810 \expandafter\in@\csname sym#3\expandafter\endcsname
811 \expandafter{\group@list}%
812 \ifin@
813 \begingroup
814 \count\z@=#4\relax
815 \count\tw@\count\z@

```

```

816 \divide\count\z@\sixt@@n
817 \count@\count\z@
818 \multiply\count@\sixt@@n
819 \advance\count\tw@-\count@
820 \if\relax\noexpand#1% is command?

```

Store the command name with a space attached inside `\reserved@@b` in case we look at a robust definition.

```

821 \edef\reserved@b{\expandafter\noexpand
822 \csname\expandafter\@gobble\string#1\space\endcsname}%

```

Test both #1<sub>U</sub> and #1<sub>L</sub> for containing `mathchar`.

```

823 \edef\reserved@a
824 {\noexpand\in@{\expandafter\@gobble\string\mathchar}%
825 {\meaning#1\expandafter\meaning\reserved@b}}%
826 \reserved@a

```

Drop #1<sub>U</sub> in case it was defined before.

```

827 \global\expandafter\let\reserved@b\undefined
828 \ifin@
829 \expandafter\set@mathsymbol
830 \csname sym#3\endcsname#1#2%
831 {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
832 \@font@info{Redeclaring math symbol \string#1}%
833 \else
834 \expandafter\ifx
835 \csname\expandafter\@gobble\string#1\endcsname
836 \relax
837 \expandafter\set@mathsymbol
838 \csname sym#3\endcsname#1#2%
839 {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
840 \else
841 \@latex@error{Command ‘\string#1’ already defined}\@eha
842 \fi
843 \fi
844 \else
845 \expandafter\set@mathchar
846 \csname sym#3\endcsname#1#2
847 {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
848 \fi
849 \endgroup
850 \else
851 \@latex@error{Symbol font ‘#3’ is not defined}\@eha
852 \fi
853 }
854 \onlypreamble\DeclareMathSymbol

```

(End definition for `\DeclareMathSymbol`.)

`\set@mathchar`

```

855 \def\set@mathchar#1#2#3#4{%
856 \global\mathcode‘#2=“\mathchar@type#3\hexnumber@#1#4\relax}
857 \onlypreamble\set@mathchar

```

(End definition for `\set@mathchar`.)

```

\set@mathsymbol
 858 \def\set@mathsymbol#1#2#3#4{%
 859 \global\mathchardef#2"\mathchar@type#3\hexnumber@#1#4\relax}
 860 \onlypreamble\set@mathsymbol

(End definition for \set@mathsymbol.)

 861 \% \def\mathsymbol#1#2#3{%
 862 % \tempcnta=#3\relax
 863 % \tempcntb\tempcnta
 864 % \divide\tempcnta\sixt@n
 865 % \count@\tempcnta
 866 % \multiply\count@\sixt@n
 867 % \advance\tempcntb-\count@
 868 % \mathchar"\mathchar@type#1\hexnumber@#2%
 869 % \hexnumber@\tempcnta\hexnumber@\tempcntb\relax}
 870 %
 871 \% \def\DeclareMathAlphabetCharacter#1#2#3{%
 872 % \DeclareMathSymbol{#1}7{#2}{#3}}

```

\DeclareMathDelimiter

```

 873 \def\DeclareMathDelimiter#1{%
 874 \if\relax\noexpand#1%
 875 \expandafter\@DeclareMathDelimiter
 876 \else
 877 \expandafter\@xxDeclareMathDelimiter
 878 \fi
 879 #1}
 880 \onlypreamble\DeclareMathDelimiter

(End definition for \DeclareMathDelimiter.)

```

\@xxDeclareMathDelimiter This macro checks if the second arg is a “math type” such as `\mathopen`. The undocumented original code didn’t use math types when the delimiter was a single letter. For this reason the coding is a bit strange as it tries to support the undocumented syntax for compatibility reasons.

```
 881 \% \def\@xxDeclareMathDelimiter#1#2#3#4{%
```

7 is the default value returned in the case that `\mathchar@type` is passed something unexpected, like a math symbol font name. We locally move `\mathalpha` out of the way so if you use that the right branch is taken. This will still fail if an explicit number 7 is used!

```

 882 \begingroup
 883 \let\mathalpha\mathord
 884 \ifnum7=\mathchar@type{#2}%
 885 \endgroup

```

If this branch is taken we have old syntax (5 arguments).

```

 886 \expandafter\@firstofone
 887 \else

```

If this branch is taken `\mathchar@type` is different from 7 so we assume new syntax. In this case we also use the arguments to set up the letter as a math symbol for the case where it is not used as a delimiter.

```

 888 \endgroup
 889 \DeclareMathSymbol{#1}{#2}{#3}{#4}%

```

Then we arrange that `\@xDeclareMathDelimiter` only gets #1, #3, #4 ... as it does not expect a math type as argument.

```

890 \expandafter\@firstoftwo
891 \fi
892 {\@xDeclareMathDelimiter#1}{#2}{#3}{#4}}
893 \onlypreamble\@xxDeclareMathDelimiter

```

(End definition for `\@xxDeclareMathDelimiter`.)

`\@DeclareMathDelimiter`

```

894 \def\@DeclareMathDelimiter#1#2#3#4#5#6{%
895 \expandafter\in@\csname sym#3\expandafter\endcsname
896 \expandafter{\group@list}%
897 \ifin@
898 \expandafter\in@\csname sym#5\expandafter\endcsname
899 \expandafter{\group@list}%
900 \ifin@
901 \begingroup
902 \count\z@=#4\relax
903 \count\tw@ \count\z@
904 \divide\count\z@\sixt@@n
905 \count@\count\z@
906 \multiply\count@\sixt@@n
907 \advance\count\tw@-\count@
908 \edef\reserved@c{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
909 %
910 \count\z@=#6\relax
911 \count\tw@ \count\z@
912 \divide\count\z@\sixt@@n
913 \count@\count\z@
914 \multiply\count@\sixt@@n
915 \advance\count\tw@-\count@
916 \edef\reserved@d{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
917 %
918 \edef\reserved@a{\noexpand\in@
919 {\expandafter\gobble\string\delimiter}{\meaning#1}}%
920 \reserved@a
921 \ifin@
922 \expandafter\set@mathdelimiter
923 \csname sym#3\expandafter\endcsname
924 \csname sym#5\endcsname#1#2%
925 \reserved@c\reserved@d
926 \@font@info{Redeclaring math delimiter \string#1}%
927 \else
928 \expandafter\ifx
929 \csname\expandafter\@gobble\string#1\endcsname
930 \relax
931 \expandafter\set@mathdelimiter
932 \csname sym#3\expandafter\endcsname
933 \csname sym#5\endcsname#1#2%
934 \reserved@c\reserved@d
935 \else
936 \@latex@error{Command '\string#1' already defined}\@eha
937 \fi

```

```

938 \fi
939 \endgroup
940 \else
941 \@latex@error{Symbol font ‘#5’ is not defined}\@eha
942 \fi
943 \else
944 \@latex@error{Symbol font ‘#3’ is not defined}\@eha
945 \fi
946 }
947 \onlypreamble\@DeclareMathDelimiter

```

(End definition for `\@DeclareMathDelimiter`.)

`\@xDeclareMathDelimiter`

```

948 \def\@xDeclareMathDelimiter#1#2#3#4#5{%
949 \expandafter\in@\csname sym#2\expandafter\endcsname
950 \expandafter{\group@list}%
951 \ifin@%
952 \expandafter\in@\csname sym#4\expandafter\endcsname
953 \expandafter{\group@list}%
954 \ifin@%
955 \begin{group}
956 \count\z@=#3\relax
957 \count\tw@\count\z@
958 \divide\count\z@\sixt@@n
959 \count@\count\z@
960 \multiply\count@\sixt@@n
961 \advance\count\tw@-\count@
962 \edef\reserved@c{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
963 %
964 \count\z@=#5\relax
965 \count\tw@\count\z@
966 \divide\count\z@\sixt@@n
967 \count@\count\z@
968 \multiply\count@\sixt@@n
969 \advance\count\tw@-\count@
970 \edef\reserved@d{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
971 \expandafter\set@mathdelimiter
972 \csname sym#2\expandafter\endcsname\csname sym#4\endcsname#1%
973 \reserved@c\reserved@d
974 \end{group}
975 \else%
976 \@latex@error{Symbol font ‘#4’ is not defined}\@eha
977 \fi
978 \else%
979 \@latex@error{Symbol font ‘#2’ is not defined}\@eha
980 \fi
981 }
982 \onlypreamble\@xDeclareMathDelimiter

```

(End definition for `\@xDeclareMathDelimiter`.)

`\set@mathdelimiter` We have to end the definition of a math delimiter like `\lfloor` with a space and not with `\relax` as we did before, because otherwise constructs involving `\abovewithdelims` will prematurely end (pr/1329)

```

983 </2ekernel>
984 <*2ekernel | latexrelease>
985 <latexrelease>\IncludeInRelease{2019/10/01}%
986 <latexrelease> {\set@mathdelimiter}{make delimiters robust}%
987 \def\set@mathdelimiter#1#2#3#4#5#6{%

```

We use `\protected` not `\MakeRobust` so that `\bigl\lceil` etc. works inside the argument of `\protected@edef`.

```

988 \protected
989 \xdef#3{\delimiter"\mathchar@type#4\hexnumber@#1#5%
990 \hexnumber@#2#6 }%
991 % \MakeRobust#3%
992 }
993 \onlypreamble\set@mathdelimiter
994 </2ekernel | latexrelease>
995 <latexrelease>\EndIncludeInRelease
996 <latexrelease>\IncludeInRelease{0000/00/00}%
997 <latexrelease> {\set@mathdelimiter}{make delimiters robust}%
998 <latexrelease>
999 <latexrelease>\def\set@mathdelimiter#1#2#3#4#5#6{%
1000 <latexrelease> \xdef#3{\delimiter"\mathchar@type#4\hexnumber@#1#5%
1001 <latexrelease> \hexnumber@#2#6 }%
1002 <latexrelease>
1003 <latexrelease>\EndIncludeInRelease
1004 <*2ekernel>

(End definition for \set@mathdelimiter.)
```

`\set@@mathdelimiter`

```

1005 \def\set@@mathdelimiter#1#2#3#4#5{%
1006 \global\delcode`#3="\hexnumber@#1#4\hexnumber@#2#5\relax}
1007 \onlypreamble\set@mathdelimiter
```

(End definition for `\set@@mathdelimiter`.)

`\DeclareMathRadical`

```

1008 \def\DeclareMathRadical#1#2#3#4#5{%
```

Below is a crude fix to make this macro work if #1 is undefined or `\relax`. Should be improved!

```

1009 \expandafter\ifx
1010 \csname\expandafter\@gobble\string#1\endcsname
1011 \relax
1012 \let#1\radical
1013 \fi
1014 \edef\reserved@a{\noexpand\in@
1015 {\expandafter\@gobble\string\radical}{\meaning#1}}%
1016 \reserved@a
1017 \ifin@
1018 \expandafter\in@\csname sym#2\expandafter\endcsname
1019 \expandafter{\group@list}%
1020 \ifin@
1021 \expandafter\in@\csname sym#4\expandafter\endcsname
1022 \expandafter{\group@list}%
1023 \ifin@
```

```

1024 \begingroup
1025 \count\z@=#3\relax
1026 \count\tw@\count\z@
1027 \divide\count\z@\sixt@on
1028 \count@\count\z@
1029 \multiply\count@\sixt@on
1030 \advance\count\tw@-\count@
1031 \edef\reserved@c{%
1032 \hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
1033 \count\z@=#5\relax
1034 \count\tw@\count\z@
1035 \divide\count\z@\sixt@on
1036 \count@\count\z@
1037 \multiply\count@\sixt@on
1038 \advance\count\tw@-\count@
1039 \edef\reserved@d{%
1040 \hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%

```

Coded inline instead of using `\set@mathradical`

```

1041 %
1042 % \expandafter\set@mathradical
1043 % \csname sym#2\expandafter\endcsname
1044 % \csname sym#4\endcsname#1%
1045 % \reserved@c\reserved@d
1046 % \xdef#1{\radical"\expandafter\hexnumber@
1047 % \csname sym#2\endcsname\reserved@c
1048 % \expandafter\hexnumber@
1049 % \csname sym#4\endcsname\reserved@d
1050 % \relax}%
1051 % \endgroup
1052 % \else
1053 % \@latex@error{Symbol font '#4' is not defined}\@eha
1054 % \fi
1055 % \else
1056 % \@latex@error{Symbol font '#2' is not defined}\@eha
1057 % \fi
1058 % \else
1059 % \@latex@error{Command '\string#1' already defined}\@eha
1060 % \fi
1061 } \onlypreamble\DeclareMathRadical

```

(End definition for `\DeclareMathRadical`.)

Definition below was wrong it contained `\delimiter!`

```

def\set@mathradical#1#2#3#4#5{%
 \xdef#3{\radical"\hexnumber@#1#4\hexnumber@#2#5\relax}}

```

`\mathalpha` just a dummy currently  
`\let\mathalpha\relax`

(End definition for `\mathalpha`.)

```

\mathchar@type
1063 \def\mathchar@type#1{%
1064 \ifodd #1\#1\else % is this non-negative number?

```

```

1065 \ifx#1\mathord 0\else
1066 \ifx#1\mathop 1\else
1067 \ifx#1\mathbin 2\else
1068 \ifx#1\mathrel 3\else
1069 \ifx#1\mathopen 4\else
1070 \ifx#1\mathclose 5\else
1071 \ifx#1\mathpunct 6\else
1072 7% % anything else is variable ord
1073 \fi
1074 \fi
1075 \fi
1076 \fi
1077 \fi
1078 \fi
1079 \fi
1080 \fi}
1081 @onlypreamble\mathchar@type

```

(End definition for `\mathchar@type`.)

`\DeclareSymbolFontAlphabet`

```

1082 \def\DeclareSymbolFontAlphabet#1#2{%
1083 \expandafter\DeclareSymbolFontAlphabet@
1084 \csname \expandafter\gobble\string#1\space\endcsname{#2}#1}
1085 @onlypreamble\DeclareSymbolFontAlphabet

```

(End definition for `\DeclareSymbolFontAlphabet`.)

`\DeclareSymbolFontAlphabet@`

```

1086 \def\DeclareSymbolFontAlphabet@#1#2#3{%

```

We use the switch `\if@tempswa` to decide if we can declare this symbol font alphabet.

```

1087 \if@tempswattrue

```

First check if #2 is known to be a symbol font

```

1088 \expandafter\in@\csname sym#2\expandafter\endcsname
1089 \expandafter{\group@list}%
1090 \ifin@%

```

Check if #1 is defined as a math alphabet defined via `\DeclareMathAlphabet`:

```

1091 \expandafter\in@\expandafter#1\expandafter{\alpha@list}%
1092 \ifin@%

```

If so remove it from the `\alpha@list` and from all math version macros.

```

1093 \font@info{Redeclaring math alphabet \string#3}%
1094 \toks@{}%
1095 \def\alpha@elt##1##2##3{%
1096 \ifx##1\else\addto@hook\toks@{\alpha@elt##1##2##3}\fi}%
1097 \alpha@list
1098 \xdef\alpha@list{\the\toks@}%

```

Now we loop over all versions and remove the math alphabet:

```

1099 \def\version@elt##1{%
1100 \begingroup
1101 \toks@{}%
1102 \def\getanddefine@fonts####1####2{%

```

```

1103 \addto@hook\toks@{\getanddefine@fonts####1####2}%
1104 \def\install@mathalphabet##1##2{%
1105 \ifx##1\else
1106 \addto@hook\toks@{\install@mathalphabet
1107 ####1{####2}}\fi}%
1108 ##1%
1109 \xdef##1{\the\toks@}%
1110 \endgroup
1111 }%
1112 \version@list
1113 \else

```

If #3 is not defined as a math alphabet check if it is defined at all:

```

1114 \expandafter\ifx
1115 \csname\expandafter\gobble\string#1\space\endcsname
1116 \relax

```

If it is undefined, fine otherwise check if it is a math alphabet defined via `\DeclareSymbolFontAlphabet`:

```

1117 \else
1118 \edef\reserved@a{%
1119 \noexpand\in@{\string\use@mathgroup}{\meaning#1}%
1120 \reserved@a
1121 \ifin@
1122 \font@info{Redeclaring math alphabet \string#3}%
1123 \else

```

Since the command #3 is defined to be something which is not a math alphabet we have to skip redefining it.

```

1124 \tempswafalse
1125 \latex@error{Command '\string#3' already defined}\@eha
1126 \fi
1127 \fi
1128 \fi
1129 \else

```

Since the symbol font is not known we better skip defining this alphabet.

```

1130 \tempswafalse
1131 \latex@error{Unknown symbol font '#2'}\@eha
1132 \fi
1133 \if@tempswa

```

When we reach this point we are allowed to define #1 to be a symbol font math alphabet. This means that we have to set it to

```
\use@mathgroup <math-settings> \sym<name>
```

The `<math-settings>` are the one for the encoding that is used in the font shape where `\sym<name>` is pointing to. This means that we have to get it from the information stored in `\group@list`. Thus we loop through that list after defining `\group@elt` in a suitable way.

```

1134 \def\group@elt##1##2{%
1135 \expandafter\ifx\csname sym#2\endcsname##1%
1136 \expandafter\reserved@a\string##2\@nil
1137 \fi}%
1138 \def\reserved@a##1##2##3\@nil{%

```

```
1139 \def\reserved@a{##2}%
1140 \group@list
1141 \toks@{\relax\ifmmode \else \non@alpherr#1\fi}%
1142 \edef#1{\the\toks@
1143 \noexpand\use@mathgroup
1144 \expandafter\noexpand\csname M@\reserved@a\endcsname
1145 \csname sym#2\endcsname}%
1146 \def#3{\protect#1}%
1147 \fi
1148 }
1149 \onlypreamble\DeclareSymbolFontAlphabet@
1150 {/2ekernel}
```

(End definition for `\DeclareSymbolFontAlphabet`.)

# File A

## ltfssini.dtx

This file contains the top level L<sup>A</sup>T<sub>E</sub>X interface to the font selection scheme commands. See other parts of the L<sup>A</sup>T<sub>E</sub>X distribution, or *The L<sup>A</sup>T<sub>E</sub>X Companion* for higher level documentation of these commands.

### 1 NFSS Initialization

Finally, there are six commands that are to be used in L<sup>A</sup>T<sub>E</sub>X and that we will therefore protect against expansion at the wrong point: \fontfamily, \fontseries, \fontshape, \fontsize, \selectfont, and \mathversion.

```
1 {*2ekernel}
```

#### 1.1 Providing math *versions*

L<sup>A</sup>T<sub>E</sub>X provides two *versions*. We call them *normal* and *bold*, respectively.

```
2 \DeclareMathVersion{normal}
3 \DeclareMathVersion{bold}
```

Now we define the standard font change commands. We don't allow the use of \rmfamily etc. in math mode.

(Actually most are now defined further down in the file.)

First the changes to another *family*:

```
4 \%{\ DeclareRobustCommand{\rmfamily
5 % {\not@math@alphabet\rmfamily\mathrm
6 % \fontfamily\rmdefault\selectfont}
7 \%{\ DeclareRobustCommand{\sffamily
8 % {\not@math@alphabet\sffamily\mathsf
9 % \fontfamily\sffamily\selectfont}
10 \%{\ DeclareRobustCommand{\ttfamily
11 % {\not@math@alphabet\ttfamily\mathtt
12 % \fontfamily\ttdefault\selectfont}
```

Then the commands changing the *series*:

```
13 \%{\ DeclareRobustCommand{\bfseries
14 % {\not@math@alphabet\bfseries\mathbf
15 % \fontseries\bfdefault\selectfont}
16 \%{\ DeclareRobustCommand{\mdseries
17 % {\not@math@alphabet\mdseries\relax
18 % \fontseries\mddefault\selectfont}
19 \%{\ DeclareRobustCommand{\upshape
20 % {\not@math@alphabet\upshape\relax
21 % \fontshape\updefault\selectfont}
```

Then the commands changing the *shape*:

```
22 \%{\ DeclareRobustCommand{\slshape
23 % {\not@math@alphabet\slshape\relax
24 % \fontshape\sldefault\selectfont}
25 \%{\ DeclareRobustCommand{\scshape
26 % {\not@math@alphabet\scshape\relax
```

```

27 \fontshape\scdefault\selectfont}
28 \DeclareRobustCommand\itshape
29 {\not@math@alphabet\itshape\mathit
30 \fontshape\itdefault\selectfont}

```

## 2 Custom series settings for main document families

This section was introduced 2020/02/02 and for now we support a full rollback (may need splitting later).

```

31 </2ekernel>
32 <*2ekernel | latexrelease>
33 <latexrelease>\IncludeInRelease{2021/11/15}%
34 <latexrelease> {\DeclareFontSeriesDefault}{Custom series}%

```

One problem with the NFSS approach of handling the series axis turned out to be that (especially with respect to “boldness”) different font families implemented different strategies. For example, with Computer Modern fonts you normally only have `bx` whereas most PostScript fonts offered only `b` but not `bx`. As a result L<sup>A</sup>T<sub>E</sub>X’s standard setting for `\bfdefault` didn’t work with such fonts, but if it got changed to produce `b`, then that didn’t work with Computer Modern if the fonts got combined (e.g., using Computer Modern Typewriter with such fonts).

The solution back then was to provide substitution rules in the font .fd such that if a `bx` series got requested the `b` series got used. While this works in that particular case, it isn’t a very general solution. For example, if you happen to have a font family that has several weights you may want to typeset the whole document in a somewhat lighter or darker font but if you then modify `\mddefault` to allow for this, then of course your change only works with that particular family but not with the typewriter or sans serif family you also want to use.

A better solution was provided by the `mweights` package by Bob Tennent that offers defaults on the level of the three main font families in the document: for “rm”, “sf” and “tt” so that font packages could define defaults for the sans serif document font by providing `\bfseries@sf` which then was used when `\bfseries` got executed and the current family was the `\sffamily`.

`\DeclareFontSeriesDefault`

We now support this concept directly from within L<sup>A</sup>T<sub>E</sub>X and for use in font packages (or the document preamble) we offer `\DeclareFontSeriesDefault`. This declaration takes three arguments:

**document family interface:** Can either be `rm`, `sf` or `tt`. This is optional and if not given the overall default.

**document series interface:** Can be `md` or `bf`.

**series value:** This is the value that is going to be used with the combination is requested.

For example, `\DeclareFontSeriesDefault[rm]{bf}{sb}` would use `sb` (semi-bold) when `\rmfamily\bfseries` is asked for.

If used without the optional argument, e.g., `\DeclareFontSeriesDefault{bf}{b}` then this is like redefining `\bfdefault` or `\mddefault`.

If some family specify defaults aren't given, e.g. if there are no declarations for, say, `tt` then the format defaults of `\mddefault` and `\bfdefault` are assumed. If those are later changed this is *not* reflected!<sup>28</sup>

`\DeclareFontSeriesDefault` The command to declare font series defaults for the “rm”, “sf” or “tt” family.

```

35 \let\DeclareFontSeriesDefault\@undefined % for rollback
36 \newcommand\DeclareFontSeriesDefault[3][]{%
37 \expandafter\def\@empty
38 \ifcsname#2series\endcsname % supported are
39 \ifcsname#3series\endcsname
40 \def\reserved@a{\#1}%

```

No optional argument: set up general default.

```

41 \ifx\reserved@a\@empty
42 \ifcsname #2series\endcsname % supported are
43 \ifcsname #3series\endcsname
44 \expandafter\def\@empty
45 \csname #2default\endcsname{\#3\@empty}%
46 \expandafter\def\@empty
47 \csname #2default@previous\endcsname{\#3\@empty}%
48 \else

```

Adding `\@empty` allows us to detect if the default gets redefined with `\renewcommand` or `\def` by the user.

```

49 \expandafter\def\@empty
50 \csname #2default\endcsname{\#3\@empty}%
51 \fi
52 \else
53 \ifcsname #2series\endcsname % supported are
54 \ifcsname #3series\endcsname
55 \expandafter\edef\@empty
56 \csname #2series\endcsname{\#3\@empty}%

```

Optional argument given, set up specific default.

```

57 \expandafter\let\@empty
58 \csname #2series\endcsname{\#1\@kernel\endcsname\@undefined}
59 \else
60 \expandafter\def\@empty
61 \csname #2series\endcsname{\#1\@kernel\endcsname\@undefined}%
62 \fi
63 \fi
64 \fi
65 }
```

---

<sup>28</sup>I see no easy way to achieve this without compromising compatibility with existing packages that currently use `mweights` and directly define (some) of the `\mdseries@..` commands but not others.

```

(End definition for \DeclareFontSeriesDefault.)
```

```

66 {/2ekernel | latexrelease}
67 <latexrelease>\EndIncludeInRelease
68 <latexrelease>\IncludeInRelease{2020/02/02}%
69 <latexrelease> {\DeclareFontSeriesDefault}{Custom series}%
70 <latexrelease>
71 <latexrelease>\let\DeclareFontSeriesDefault\@undefined % for rollback
72 <latexrelease>\newcommand\DeclareFontSeriesDefault[3] []{%
73 <latexrelease> \def\reserved@a{\#1}%
74 <latexrelease> \ifx\reserved@a\empty%
75 <latexrelease> \ifcsname #2series\endcsname % supported are
76 <latexrelease> % \[md/bf]default
77 <latexrelease> \expandafter\def
78 <latexrelease> \csname #2default\endcsname{\#3\empty}%
79 <latexrelease> \expandafter\def
80 <latexrelease> \csname #2default@previous\endcsname{\#3\empty}%
81 <latexrelease> \else
82 <latexrelease> \@latex@error{Wrong syntax for \string\DeclareFontSeriesDefault}%
83 <latexrelease> {Mandatory first argument must be 'md' or 'bf'.}
84 <latexrelease> \fi
85 <latexrelease> \else
86 <latexrelease> \ifcsname #2series@#1\endcsname % supported are
87 <latexrelease> % \[md/bf]series@[rm/sf/tt]
88 <latexrelease> \expandafter\edef
89 <latexrelease> \csname #2series@#1\endcsname{\#3}%
90 <latexrelease> \expandafter\let
91 <latexrelease> \csname #2series@#1@kernel\endcsname\@undefined
92 <latexrelease> \else
93 <latexrelease> \@latex@error{Wrong syntax for \string\DeclareFontSeriesDefault}%
94 <latexrelease> {Optional argument must be 'rm', 'sf', or 'tt'. \MessageBreak
95 <latexrelease> Mandatory first argument must be 'md' or 'bf'.}
96 <latexrelease> \fi
97 <latexrelease> \fi
98 <latexrelease> }
99 <latexrelease>
100 <latexrelease>\EndIncludeInRelease
101 <latexrelease>\IncludeInRelease{0000/00/00}%
102 <latexrelease> {\DeclareFontSeriesDefault}{Custom series}%
103 <latexrelease>
104 <latexrelease>\let\DeclareFontSeriesDefault\@undefined
105 <latexrelease>\let\bfseries@rm\@undefined
106 <latexrelease>\let\bfseries@sf\@undefined
107 <latexrelease>\let\bfseries@tt\@undefined
108 <latexrelease>\let\bfseries@rm@kernel\@undefined
109 <latexrelease>\let\bfseries@sf@kernel\@undefined
110 <latexrelease>\let\bfseries@tt@kernel\@undefined
111 <latexrelease>\let\mdseries@rm\@undefined
112 <latexrelease>\let\mdseries@sf\@undefined
113 <latexrelease>\let\mdseries@tt\@undefined
114 <latexrelease>\expandafter\let\csname ver@mweights.sty\endcsname\@undefined
115 <latexrelease>
116 <latexrelease>\let\@meta@family@list\@undefined
117 <latexrelease>\let\prepare@family@series@update\@undefined
118 <latexrelease>\let@update@series@target@value\@undefined

```

```
119 \begin{macro}{\textrm{}}{}
```

This is always called in `\document` so don't make it undefined.

```
120 \begin{macro}{\textrm}{\relax}
```

```
121 \begin{macro}{\textrm}{\textrm}
```

```
122 \begin{macro}{\textrm}{\EndIncludeInRelease}
```

```
123 \begin{macro}{\textrm}{\relax}
```

```
124 \begin{macro}{\textrm}{\relax}
```

```
125 \begin{macro}{\textrm}{\textrm}
```

```
126 \begin{macro}{\textrm}{\IncludeInRelease{2020/02/02}}
```

```
127 \begin{macro}{\textrm}{\textrm}
```

```
 {\mdseries@rm}{Custom series}}
```

`\mdseries@rm` We initialize the family specific default at the end of the format generation. Later on they may get overwritten in the preamble or a package via `\DeclareFontSeriesDefault` (or possibly directly).

`\bfseries@rm` Conceptual change: The `\bfdefault` will be `b` not `bx` because that is what it should be really for nearly every font except Computer/Latin Modern.

`\bfseries@sf` To account for the fact that by default we typeset in CM or LM we set up the `\bfseries@..` defaults to use `bx` instead.

This means that it behaves like before because if the default fonts are used then `\bfseries@rm` etc kick in and make `\textbf` use `bx`. However, if the font gets changed then `\bfdefault` will get used.

```
128 \def\bfsf@rm{bx}
```

```
129 \def\bfsf@sf{bx}
```

```
130 \def\bfsf@tt{bx}
```

Frozen version of the kernel defaults so we can see if they have changed.

```
131 \let\bfsf@rm@kernel\bfsf@rm
```

```
132 \let\bfsf@sf@kernel\bfsf@sf
```

```
133 \let\bfsf@tt@kernel\bfsf@tt
```

The default for the medium series is `m` and this will be interpreted as resetting both weight and width. To reset only one of them the virtual value `?m` and `m?` are available.

```
134 \def\mdseries@rm{m}
```

```
135 \def\mdseries@sf{m}
```

```
136 \def\mdseries@tt{m}
```

(End definition for `\mdseries@rm` and others.)

`\series@change@debug` For debugging, but right now none of this code is extracted. The idea is to have a separate package with debugging code one day.

```
137 \let\series@change@debug\typeout
```

```
138 \let\series@change@debug\@gobble
```

```
139 \let\series@change@debug\@empty
```

```
140 \end{macro}
```

(End definition for `\series@change@debug`.)

`\prepare@family@series@update` This is core command that prepares for the family update. The big difference to the documented code above is that the nested `\ifx` statements seem to be missing. Instead we loop through an internal list that holds the names of the three meta families. This approach allows us to extend the mechanism at a later stage to allow for additional named meta families.

Here is the current definition of that list:

```
141 \def\@meta@family@list{\@elt{rm}\@elt{sf}\@elt{tt}}
```

```
142 \def\prepare@family@series@update#1#2{%
```

```
143 \if@forced@series
```

```
144 <+debug> \series@change@debug{No series preparation (forced \f@series)\on@line}%
145 \fontfamily#2%
```

```
146 \else
```

```
147 <+debug> \series@change@debug{Preparing for switching to #1 (#2)\on@line}%
148 \expand@font@defaults
```

We prepare for changing the current series. We have to find it before changing the family as discussed above.

```
149 \let\target@series@value\empty
150 \def\target@meta@family@value{#1}%
```

As the very last item in the meta family list we add `\@elt{??}` and define this pseudo meta family to be the current font family. So if none of the real meta families matched then this will match. This will cover the following case:

- `\bfseries` is called for a family using `bx` (e.g., CMR)
- Switch to a font family that is none of the meta families, e.g., via `\fontfamily{ptm}\allowbreak\verb+\bfseries+`
- Then none of the real meta families, match but the final `\@elt{??}` will.
- Therefore if the current series is `\mddefault` or `\bfdefault` it will be detected and the corresponding target series selected.

```
151 \expandafter\edef\csname ??def@ult\endcsname{\f@family}%
```

To find it we loop over the meta family list with a suitable definition of `\@elt`.

```
152 \let\@elt\update@series@target@value
153 \@meta@family@list
```

Last resort pseudo meta family. Will only be looked at if none of the real ones have matched.

```
154 \@elt{??}%
155 \let\@elt\relax
```

That will figure out the correct series value to use without updating it. Now we can change the family.

```
156 \fontfamily#2%
```

After that we update the series. That code is again like the one above.

```
157 \ifx\target@series@value\empty
158 <+debug> \series@change@debug{Target series still empty ...}%
159 \else
160 \ifx\f@series\target@series@value
161 <+debug> \series@change@debug{Target series unchanged:
162 <+debug> \f@series \space = \target@series@value}%
163 \else
164 \maybe@load@fontshape
165 <+debug> \series@change@debug{Target series:
166 <+debug> \f@series \space -> \target@series@value}%
```

The `\target@series@value` may contain something like `cm` (coming from a default) and so we can't directly assign it to `\f@series` because we have to drop any surplus `m` first.

```

167 % \let\f@series\target@series@value
168 \series@maybe@drop@one@m\target@series@value\f@series
169 \fi
170 \fi
171 \fi
172 }
```

(End definition for `\prepare@family@series@update` and `\@meta@family@list`.)

#### `\update@series@target@value`

In this macro used in the loop you basically find the nested `\ifxs` from the outline above. The only difference is that it is parameterized instead of being written out and only for one block of tests because the code is called repeatedly when looping over the meta family list. From the list we get each meta family name in turn.

```
173 \def\update@series@target@value#1{%
```

There is one additional test at the beginning, because the list contains all meta families and we need to ignore the case where current one from the list and target one are identical.

```

174 \def\reserved@a{\#1}%
175 \ifx\target@meta@family@value\reserved@a % rm -> rm do nothing
176 \else
177 {+debug} \series@change@debug{Trying to match #1: \csname#1\def@ult\endcsname
178 {+debug} \space = \f@family\space ?}%

```

We only "do" something if the current font family matches the current meta family.

```
179 \expandafter\ifx\csname#1\def@ult\endcsname\f@family
```

If that's the case we know that this is the block that applies (only one meta family can match). So to speed things up we change `\@elt` so that the rest of the loop gets gobbled.

```
180 \let\@elt\@gobble
```

Then we try to find the right new value for the series (as explained above). The two macros defined first are only there because we now need to use `\csname` and this way the code will be a little faster.

```

181 \expandafter\let\expandafter\reserved@b
182 \csname mdseries@\target@meta@family@value\endcsname
183 \expandafter\let\expandafter\reserved@c
184 \csname bfseries@\target@meta@family@value\endcsname
185 {+debug} \series@change@debug{Targets for mdseries and bfseries:
186 {+debug} \reserved@b\space and \reserved@c}%

```

This here is now identical to the nested `\ifx` block from the outline, except that it there appeared twice in `\rmfamily`. This is now covered by looping and stopping the loop when a match was found.

We have to sanitize the default value first because it may contain something like `mc` and that would never match `\f@series` because there it would be called `c` with the `m` dropped. It would be probably better to do that differently these days, but it is hard to adjust without causing a lot of issues, so we do the dropping in various places instead.

```

187 \expandafter\series@maybe@drop@one@m
188 \csname mdseries@#1\endcsname\reserved@d
189 \ifx\reserved@d\f@series
190 {+debug} \series@change@debug{mdseries@#1 matched -> \reserved@b}%
191 \let\target@series@value\reserved@b
192 \else

```

Again do some sanitizing.

```
193 \expandafter\series@maybe@drop@one@m
194 \csname bfseries@#1\endcsname\reserved@d
195 \ifx\reserved@d\f@series
196 {+debug} \series@change@debug{bfseries@#1 matched -> \reserved@c}%
197 \let\target@series@value\reserved@c
198 \else\ifx\f@series\mddef@ult \let\target@series@value\reserved@b
199 {+debug} \series@change@debug{mddef@ult matched -> \reserved@b}%
200 \else\ifx\f@series\bfdef@ult \let\target@series@value\reserved@c
201 {+debug} \series@change@debug{bfdef@ult matched -> \reserved@c}%
202 \fi\fi\fi\fi
203 \fi
204 \fi
205 }
```

(End definition for \update@series@target@value.)

\init@series@setup This is code to be run at begin document ...

```
206 \def\init@series@setup{%
```

We only want **bx** in \bfseries@rm if the roman font is Computer Modern or Latin Modern, otherwise it should be **b**. It was set to **bx** in the kernel so that any font use with the default families in the preamble get this value. Now at the real document start we check if the fonts have been changed. If there was a \DeclareFontSeriesDefault declaration or \bfseries@rm was directly altered then it differs from \bfseries@rm@kernel and we do nothing. Otherwise we check if \rmdefault is one of the CM/LM font families and if so we keep **bx** otherwise we change it to **b**.

This approach doesn't cover one case: CM/LM got changed to a different family that supports **bx**, but the support package for that family used \def\bfseries@rm{bx} instead of using \DeclareFontSeriesDefault. In that case the code here changes it to **b**. Solution: use the \DeclareFontSeriesDefault interface.

```
207 \ifx\bfseries@rm@kernel\bfseries@rm
208 \expandafter\in@\expandafter{\rmdefault}%
209 {cmr,cmss,cmtt,lcms,lcmtt,lmr,lmss,lmmt}%
210 \ifin@ \else \def\bfseries@rm{b}\fi\fi
```

Same approach for \bfseries@sf and \bfseries@tt:

```
211 \ifx\bfseries@sf@kernel\bfseries@sf
212 \expandafter\in@\expandafter{\sfdefault}%
213 {cmr,cmss,cmtt,lcms,lcmtt,lmr,lmss,lmmt}%
214 \ifin@ \else \def\bfseries@sf{b}\fi\fi
215 \ifx\bfseries@tt@kernel\bfseries@tt
216 \expandafter\in@\expandafter{\ttdefault}%
217 {cmr,cmss,cmtt,lcms,lcmtt,lmr,lmss,lmmt}%
218 \ifin@ \else \def\bfseries@tt{b}\fi\fi
```

If the document preamble has changed the \familydefault or if the if the \rmdefault contains a new font family, we may have to adjust the series defaults accordingly, before starting typesetting.

Similarly, if the user has changed the \mddefault or the medium series for the family selected as document font we may also have to adjust the \seriesdefault.

On the other hand if the document font is still CM or LM then \bfdefault is wrong, because it is now saying **b** and not **bx** as it should for such fonts.

To fix all this we first run `\reset@font` (the internal kernel name for `\normalfont`). This will set up the document encoding, family, series and shape based on the current values of `\encodingdefault`, `\familydefault`, `\seriesdefault` and `\shapedefault`. However, if the family (from `\familydefault`) has special medium default we should switch to that (and not use what is current value from `\seriesdefault`). This can be achieved by afterwards calling `\mediumseries` and then changing `\seriesdefault` to the now current series value (in `\f@series`).

But what should happen if `\seriesdefault` got explicitly changed? In that case the explicit change should survive and we should not alter `\seriesdefault`. This is solved by comparing the current value of `\seriesdefault` with a kernel version saved in the format and if they differ we do not call `\mdseries` or change `\seriesdefault`.

```

219 \reset@font
220 \ifx\seriesdefault\seriesdefault@kernel
221 \mdseries
222 \let\seriesdefault\f@series
223 \fi
224 }%

```

*(End definition for `\init@series@setup`.)*

As the kernel code now implements the same functionality as `mweights`, albeit internally coded slightly differently, that package shouldn't be loaded any more. We therefore pretend that it already got loaded. Thus, a font package that tries to load it and then sets `\mdseries@...`, etc. will continue to work but will now use the kernel code.

Of course, mid-term such package should probably use `\DeclareFontSeriesDefault` instead of making using low-level definitions.

```

225 \expandafter\let\csname ver@mweights.sty\endcsname\fmtversion
226 {/2ekernel | latexrelease}
227 \langle latexrelease\rangle\EndIncludeInRelease
228 \langle latexrelease\rangle\IncludeInRelease{0000/00/00}%
229 \langle latexrelease\rangle\mdseries@rm-{Custom series}%
230 \langle latexrelease\rangle
231 \langle latexrelease\rangle\let\bfseries@rm@\undefined
232 \langle latexrelease\rangle\let\bfseries@sf@\undefined
233 \langle latexrelease\rangle\let\bfseries@tt@\undefined
234 \langle latexrelease\rangle\let\bfseries@rm@kernel@\undefined
235 \langle latexrelease\rangle\let\bfseries@sf@kernel@\undefined
236 \langle latexrelease\rangle\let\bfseries@tt@kernel@\undefined
237 \langle latexrelease\rangle\let\mdseries@rm@\undefined
238 \langle latexrelease\rangle\let\mdseries@sf@\undefined
239 \langle latexrelease\rangle\let\mdseries@tt@\undefined
240 \langle latexrelease\rangle\expandafter\let\csname ver@mweights.sty\endcsname@\undefined
241 \langle latexrelease\rangle
242 \langle latexrelease\rangle\let\@meta@family@list@\undefined
243 \langle latexrelease\rangle\let\prepare@family@series@update@\undefined
244 \langle latexrelease\rangle\let\update@series@target@value@\undefined
245 \langle latexrelease\rangle

```

This is always called in `\document` so don't make it undefined.

```

246 \langle latexrelease\rangle\let\init@series@setup\relax
247 \langle latexrelease\rangle
248 \langle latexrelease\rangle\EndIncludeInRelease
249 {*2ekernel}

```

```

250 </2ekernel>
251 <*2ekernel | latexrelease>
252 <latexrelease>\IncludeInRelease{2021/11/15}%
253 <latexrelease> {\bfseries}{Custom series with hooks}%

```

\bfseries This document command switches to the bold series.

```

254 \DeclareRobustCommand{\bfseries}{%
255 \not@math@alphabet\bfseries\mathbf

```

In the original NFSS definition it then called \fontseries with the value \bfdefault. In the new scheme we have more alternatives and therefore check if the current family (\f@family) is the current \rmdef@ult, \sfdef@ult or \ttdef@ult and the select the correct family default in that case.

```

256 \expand@font@defaults
257 \maybe@update@bfseries@defaults
258 \ifx\f@family\rmdef@ult \fontseries\bfseries@rm
259 \else\ifx\f@family\sfdef@ult \fontseries\bfseries@sf
260 \else\ifx\f@family\ttdef@ult \fontseries\bfseries@tt

```

If not \bfdefault is used.

```

261 \else \fontseries\bfdefault
262 \fi\fi\fi

```

This hook in contrast is always executed.

```

263 \UseHook{\bfseries}%
264 \selectfont
265 }

```

(End definition for \bfseries.)

\maybe@update@bfseries@defaults If \bfdefault and \bfdefault@previous are different then the default got changed directly through the legacy interface (i.e., via \def or \renewcommand. In that case we reset all meta family defaults so that the document behaves like it was the case before the new mechanism was introduced.

```

266 \def\maybe@update@bfseries@defaults{%
267 \ifx\bfdefault\bfdefault@previous\else

```

We add \@empty and then let \bfdefault@previous to \bfdefault so that we can detect any further change.

```

268 \expandafter\def\expandafter\bfdefault
269 \expandafter{\bfdefault\@empty}%
270 \let\bfdefault@previous\bfdefault

```

And we reset the meta family defaults (\bfdef@ult is an expanded version of \bfdefault.

```

271 \let\bfseries@rm\bfdef@ult
272 \let\bfseries@sf\bfdef@ult
273 \let\bfseries@tt\bfdef@ult

```

Formats that set up parallel fonts, e.g., for Japanese, can use this hook to add resets here. Note that this hook is only run when resets are necessary.

```

274 \UseHook{\bfseries/defaults}%
275 \fi
276 }

```

(End definition for \maybe@update@bfseries@defaults.)

\mdseries This document command switches to the medium series.

```
277 \DeclareRobustCommand\mdseries{%
278 \not@math@alphabet\mdseries\relax
279 \expand@font@defaults
280 \maybe@update@\mdseries@defaults
281 \ifx\f@family\rmdef@ult \fontseries\mdseries@rm
282 \else\ifx\f@family\sfdef@ult \fontseries\mdseries@sf
283 \else\ifx\f@family\ttdef@ult \fontseries\mdseries@tt
284 \else \fontseries\mddefault
285 \fi\fi\fi
286 \UseHook{\mdseries}%
287 \selectfont
288 }
```

(End definition for \mdseries.)

\maybe@update@\mdseries@defaults

```
289 \def\maybe@update@\mdseries@defaults{%
290 \ifx\mddefault\mddefault@previous\else
291 \expandafter\def\expandafter\mddefault\expandafter{\mddefault\empty}%
292 \let\mddefault@previous\mddefault
293 \let\mdseries@rm\mddef@ult
294 \let\mdseries@sf\mddef@ult
295 \let\mdseries@tt\mddef@ult
```

Formats that set up parallel fonts, e.g., for Japanese, can use this hook to add resets here.

```
296 \UseHook{\mdseries/defaults}%
297 \fi
298 }
```

(End definition for \maybe@update@\mdseries@defaults.)

```
299 </2ekernel | latexrelease>
300 <latexrelease>\EndIncludeInRelease
301 <latexrelease>\IncludeInRelease{2020/10/01}%
302 <latexrelease> {\bfseries}{Custom series with hooks}%
303 <latexrelease>
304 <latexrelease>\let\maybe@update@bfseries@defaults\undefined
305 <latexrelease>\let\maybe@update@\mdseries@defaults\undefined
306 <latexrelease>
307 <latexrelease>\DeclareRobustCommand\bfseries{%
308 <latexrelease> \not@math@alphabet\bfseries\mathbf
309 <latexrelease> \expand@font@defaults
310 <latexrelease> \ifx\bfdefault\bfdefault@previous\else
311 <latexrelease> \expandafter\def\expandafter\bfdefault
312 <latexrelease> \expandafter{\bfdefault\empty}%
313 <latexrelease> \let\bfdefault@previous\bfdefault
314 <latexrelease> \let\bfseries@rm\bfdef@ult
315 <latexrelease> \let\bfseries@sf\bfdef@ult
316 <latexrelease> \let\bfseries@tt\bfdef@ult
317 <latexrelease> \UseHook{\bfseries/defaults}%
318 <latexrelease> \fi
319 <latexrelease> \ifx\f@family\rmdef@ult \fontseries\bfseries@rm
320 <latexrelease> \else\ifx\f@family\sfdef@ult \fontseries\bfseries@sf
```

```

321 〈latexrelease〉 \else\ifx\f@family\ttdef@ult \fontseries\bfseries@tt
322 〈latexrelease〉 \else \fontseries\bfdefault
323 〈latexrelease〉 \fi\fi\fi
324 〈latexrelease〉 \UseHook{bfseries}%
325 〈latexrelease〉 \selectfont
326 〈latexrelease〉}
327 〈latexrelease〉
328 〈latexrelease〉\DeclareRobustCommand\mdseries{%
329 〈latexrelease〉 \not@math@alphabet\mdseries\relax
330 〈latexrelease〉 \expand@font@defaults
331 〈latexrelease〉 \ifx\mddefault\mddefault@previous\else
332 〈latexrelease〉 \expandafter\def\expandafter\mddefault\expandafter{\mddefault\emptyset}%
333 〈latexrelease〉 \let\mddefault@previous\mddefault
334 〈latexrelease〉 \let\mdseries@rm\mddef@ult
335 〈latexrelease〉 \let\mdseries@sf\mddef@ult
336 〈latexrelease〉 \let\mdseries@tt\mddef@ult
337 〈latexrelease〉 \UseHook{mdseries/defaults}%
338 〈latexrelease〉 \fi
339 〈latexrelease〉 \ifx\f@family\rmdef@ult \fontseries\mdseries@rm
340 〈latexrelease〉 \else\ifx\f@family\sfdef@ult \fontseries\mdseries@sf
341 〈latexrelease〉 \else\ifx\f@family\ttdef@ult \fontseries\mdseries@tt
342 〈latexrelease〉 \else \fontseries\mddefault
343 〈latexrelease〉 \fi\fi\fi
344 〈latexrelease〉 \UseHook{mdseries}%
345 〈latexrelease〉 \selectfont
346 〈latexrelease〉}
347 〈latexrelease〉\EndIncludeInRelease
348 〈latexrelease〉\IncludeInRelease{2020/02/02}%
349 〈latexrelease〉 {\bfseries}{Custom series with hooks}%
350 〈latexrelease〉
351 〈latexrelease〉
352 〈latexrelease〉\DeclareRobustCommand\bfseries{%
353 〈latexrelease〉 \not@math@alphabet\bfseries\mathbf
354 〈latexrelease〉 \expand@font@defaults
355 〈latexrelease〉 \ifx\f@family\rmdef@ult \fontseries\bfseries@rm
356 〈latexrelease〉 \else\ifx\f@family\sfdef@ult \fontseries\bfseries@sf
357 〈latexrelease〉 \else\ifx\f@family\ttdef@ult \fontseries\bfseries@tt
358 〈latexrelease〉 \else \fontseries\bfdefault
359 〈latexrelease〉 \fi\fi\fi
360 〈latexrelease〉 \selectfont
361 〈latexrelease〉}
362 〈latexrelease〉
363 〈latexrelease〉\DeclareRobustCommand\mdseries{%
364 〈latexrelease〉 \not@math@alphabet\mdseries\relax
365 〈latexrelease〉 \expand@font@defaults
366 〈latexrelease〉 \ifx\f@family\rmdef@ult \fontseries\mdseries@rm
367 〈latexrelease〉 \else\ifx\f@family\sfdef@ult \fontseries\mdseries@sf
368 〈latexrelease〉 \else\ifx\f@family\ttdef@ult \fontseries\mdseries@tt
369 〈latexrelease〉 \else \fontseries\mddefault
370 〈latexrelease〉 \fi\fi\fi
371 〈latexrelease〉 \selectfont
372 〈latexrelease〉}
373 〈latexrelease〉
374 〈latexrelease〉

```

```

375 〈\latexrelease〉
376 〈\latexrelease〉\EndIncludeInRelease
377 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
378 〈\latexrelease〉 {\bfseries}{Custom series with hooks}%
379 〈\latexrelease〉
380 〈\latexrelease〉\DeclareRobustCommand\bfseries
381 〈\latexrelease〉 {\not@math@alphabet\bfseries\mathbf}
382 〈\latexrelease〉 \fontseries\bfdefault\selectfont
383 〈\latexrelease〉\DeclareRobustCommand\mdseries
384 〈\latexrelease〉 {\not@math@alphabet\mdseries\relax
385 〈\latexrelease〉 \fontseries\mddefault\selectfont}
386 〈\latexrelease〉
387 〈\latexrelease〉\EndIncludeInRelease
388 {*2ekernel}
389
390
391
392
393 〈/2ekernel〉
394 {*2ekernel | \latexrelease}
395 〈\latexrelease〉\IncludeInRelease{2020/10/01}%
396 〈\latexrelease〉 {\expand@font@defaults}{Custom series with hooks}%

```

`\expand@font@defaults` The family specific defaults are fully expanded, i.e., they are defined via `\edef` inside `\DeclareFontSeriesDefault`. However, the overall defaults, e.g., `\bfdefault` may have been redefined by the user and thus may not be fully expanded. So to enable reliable comparison we make expanded versions of them. That we rerun each time. The alternative would be to only allow for changes before begin document.

`\rm@def@ult` 397 `\def\expand@font@defaults{%`

`\sf@def@ult` 398 `\edef\rmdef@ult{\rmdefault}%`

`\tt@def@ult` 399 `\edef\sfdef@ult{\sfdefault}%`

`\md@def@ult` 400 `\edef\ttdef@ult{\ttdefault}%`

The series defaults may contain some surplus `m` that we need to drop here.

```

401 \series@maybe@drop@one@m\bfdefault\bfdef@ult
402 \series@maybe@drop@one@m\mddefault\mddef@ult

```

Formats that set up parallel fonts, e.g., for Japanese, can use this hook to add additional code here.

```

403 \UseHook{\expand@font@defaults}%
404 }

```

(End definition for `\expand@font@defaults` and others.)

`\rmfamily` Here are the document level commands for changing the main font families, or rather, here is a documented outline of the code, the actual code is then streamlined and somewhat generalized.

```

\rmfamily\rmfamily{%
\not@math@alphabet\rmfamily\mathrm

```

If families are changed then we have to do a bit more work. In the original NFSS implementation a family change kept encoding, series shape and size unchanged but now we can't any longer simply reuse the current series value. Instead we may have to change it from one family default to the next.

```
\expand@font@defaults
```

We have to do the testing while the current family is still unchanged but we have to do the adjustment of the series after it got changed (because the new family might have different sets of shapes available and we certainly don't want to see substitution going on. So we use `\target@series@value` to hold the target series (if any).

```
\let\target@series@value\empty
```

Thus, if the current family is the sans family

```
\ifx\f@family\sfdef@ult
```

and if we using the medium series of the sans family

```
\ifx\f@series\mdseries@sf
```

then lets switch to the medium series for the serif family

```
\let\target@series@value\mdseries@rm
```

and if we use the bold series of the sans family switch to the bold default of the serif family:

```
\else\ifx\f@series\bfseries@sf \let\target@series@value\bfseries@rm
```

However, the sans family may not have any specific defaults set, so we also compare with the overall defaults.

```
\else\ifx\f@series\mddef@ult \let\target@series@value\mdseries@rm
\else\ifx\f@series\bfdef@ult \let\target@series@value\bfseries@rm
```

If neither test was true we leave the series alone. This way a special manual setting such as `\fontseries{lc}` is not undone if the family changes (of course there may not be any support for it in the new family but then the NFSS substitution kicks in and sorts it out).

```
\fi\fi\fi\fi
```

We need to do the same if the current family is the typewriter family:

```
\else\ifx\f@family\ttdef@ult
 \ifx\f@series\mdseries@tt \let\target@series@value\mdseries@rm
 \else\ifx\f@series\bfseries@tt \let\target@series@value\bfseries@rm
 \else\ifx\f@series\mddef@ult \let\target@series@value\mdseries@rm
 \else\ifx\f@series\bfdef@ult \let\target@series@value\bfseries@rm
 \fi\fi\fi\fi
\fi\fi
```

With these preparations for series out of the way we can now change the font family to `\rmdefault`.

```
\fontfamily\rmdefault
```

If `\target@series@value` is still empty there is nothing more to do other than selecting the new family. However, if not then we should update the font series now as well. But there is one further subtle issue. We may not have loaded an `.fd` file for our target font family yet. In the past that was done in `\selectfont` if necessary but since we are now doing all the comparisons in `\fontseries` we need to make sure that the font family specifications are already loaded prior to calling `\fontseries`.

```
\ifx\target@series@value\empty \else
 \maybe@load@fontshape
```

Updating the series in this case means directly changing `\f@series` to the target value. We don't want to go through `\fontseries` because that would apply the mappings and then `bx + b` would keep `bx` instead of changing to `b` as desired. as

```
\let\f@series\target@series@value
\fi
\selectfont}
```

So now for the real definition: most of the code above gets delegated to a helper command `\prepare@family@series@update` so that the definition becomes again fairly short. In addition we add a hook, mainly for our Japanese friends so that the code can be extended prior to the call to `\selectfont`.

```
405 \DeclareRobustCommand\rmfamily{%
 406 \not@math@alphabet\rmfamily\mathrm}
```

This holds all the code discussed above, first argument is the meta family, i.e., `rm` in this case, and second argument is the default family name, e.g., `cmr` indirectly accessed via `\rmdefault`. This is calling `\fontfamily` and if necessary `\fontseries` as outline above.

```
407 \prepare@family@series@update{rm}\rmdefault
```

Then comes the hook code (by default a no-op) and finally the call to `\selectfont`.

```
408 \UseHook{rmfamily}%
 409 \selectfont
```

The definitions for `\sffamily` and `\ttfamily` are similar, the differences are only in what font families get checked.

```
\ttfamily 410 \DeclareRobustCommand\sffamily{%
 411 \not@math@alphabet\sffamily\mathsf
 412 \prepare@family@series@update{sf}\sfdefault
 413 \UseHook{sffamily}%
 414 \selectfont

 415 \DeclareRobustCommand\ttfamily{%
 416 \not@math@alphabet\ttfamily\mathtt
 417 \prepare@family@series@update{tt}\ttdefault
 418 \UseHook{ttfamily}%
 419 \selectfont}
```

*(End definition for `\rmfamily`, `\sffamily`, and `\ttfamily`.)*

```
rmfamily Declare the hooks used above.
sffamily 420 \NewHook{rmfamily}
ttfamily 421 \NewHook{sffamily}
normalfont 422 \NewHook{ttfamily}
expand@font@defaults 423 \NewHook{normalfont}
bfseries 424 \NewHook{expand@font@defaults}
bfseries/defaults 425 \NewHook{bfseries}
mdseries 426 \NewHook{bfseries/defaults}
mdseries/defaults 427 \NewHook{mdseries}
428 \NewHook{mdseries/defaults}
```

(End definition for `rmfamily` and others.)

`\@rmfamilyhook` These four hooks have legacy versions used in 2020/02/02 so we should support them  
`\@sffamilyhook` until they aren't any longer used.

By default the hooks do nothing

```
\@defaultfamilyhook 429 \let\@rmfamilyhook\@empty
430 \let\@sffamilyhook\@empty
431 \let\@ttfamilyhook\@empty
432 \let\@defaultfamilyhook\@empty %FMi sort out
```

(End definition for \@rmfamilyhook and others.)

```
433 </2ekernel | latexrelease>
434 <|latexrelease>\EndIncludeInRelease
435 <|latexrelease>\IncludeInRelease{2020/02/02}%
436 <|latexrelease> {\expandafter\font@defaults}{Custom series with hooks}%
437 <|latexrelease>
438 <|latexrelease>\def\expand@font@defaults{%
439 <|latexrelease> \edef\rmdef@ult{\rmdefault}%
440 <|latexrelease> \edef\sffdef@ult{\sfdefault}%
441 <|latexrelease> \edef\ttdef@ult{\ttdefault}%
442 <|latexrelease> \edef\bfdef@ult{\bfdefault}%
443 <|latexrelease> \edef\mddef@ult{\mddefault}%
444 <|latexrelease> \edef\famdef@ult{\familydefault}%
445 <|latexrelease>}
446 <|latexrelease>
447 <|latexrelease>
448 <|latexrelease>\DeclareRobustCommand\rmfamily{%
449 <|latexrelease> \not@math@alphabet\rmfamily\mathrm
450 <|latexrelease> \prepare@family@series@update{\rm}\rmdefault
451 <|latexrelease> \rmfamilyhook
452 <|latexrelease> \selectfont}
453 <|latexrelease>\DeclareRobustCommand\sffamily{%
454 <|latexrelease> \not@math@alphabet\sffamily\mathsf
455 <|latexrelease> \prepare@family@series@update{\sf}\sfdefault
456 <|latexrelease> \sffamilyhook
457 <|latexrelease> \selectfont}
458 <|latexrelease>\DeclareRobustCommand\ttfamily{%
459 <|latexrelease> \not@math@alphabet\ttfamily\mathtt
460 <|latexrelease> \prepare@family@series@update{\tt}\ttdefault
461 <|latexrelease> \ttfamilyhook
462 <|latexrelease> \selectfont}
463 <|latexrelease>\let\rmfamilyhook\empty
464 <|latexrelease>\let\sffamilyhook\empty
```

```

465 〈latexrelease〉\let\@ttfamilyhook\@empty
466 〈latexrelease〉
467 〈latexrelease〉
468 〈latexrelease〉\EndIncludeInRelease
469 〈latexrelease〉\IncludeInRelease{0000/00/00}%
470 〈latexrelease〉 {\expand@font@defaults}{Custom series with hooks}%
471 〈latexrelease〉
472 〈latexrelease〉\let\expand@font@defaults\@undefined
473 〈latexrelease〉
474 〈latexrelease〉\DeclareRobustCommand\bfseries
475 〈latexrelease〉 {\not@math@alphabet\bfseries\mathbf}
476 〈latexrelease〉 \fontseries\bfdefault\selectfont
477 〈latexrelease〉\DeclareRobustCommand\mdseries
478 〈latexrelease〉 {\not@math@alphabet\mdseries\relax}
479 〈latexrelease〉 \fontseries\mddefault\selectfont
480 〈latexrelease〉\DeclareRobustCommand\rmfamily
481 〈latexrelease〉 {\not@math@alphabet\rmfamily\mathrm}
482 〈latexrelease〉 \fontfamily\rmdefault\selectfont
483 〈latexrelease〉\DeclareRobustCommand\sffamily
484 〈latexrelease〉 {\not@math@alphabet\sffamily\mathsf}
485 〈latexrelease〉 \fontfamily\sfdefault\selectfont
486 〈latexrelease〉\DeclareRobustCommand\ttfamily
487 〈latexrelease〉 {\not@math@alphabet\ttfamily\mathtt}
488 〈latexrelease〉 \fontfamily\ttdefault\selectfont
489 〈latexrelease〉
490 〈latexrelease〉\let\@rmfamilyhook\@undefined
491 〈latexrelease〉\let\@sffamilyhook\@undefined
492 〈latexrelease〉\let\@ttfamilyhook\@undefined
493 〈latexrelease〉
494 〈latexrelease〉\EndIncludeInRelease
495 〈*2ekernel〉

```

`\IfFontSeriesContextTF` With the ability for `\bfseries` or `\mdseries` to be mapped to different NFSS axis values it becomes important to have the ability to determine the current context as we can no longer look at `\f@series` to answer a question such as “am I currently typesetting in a bold typeface?”

This is provided by the test `\IfFontSeriesContextTF`. It takes three arguments:

- The context we try to check (either `bf` for bold or `md` for medium, i.e., the same that can go into the first mandatory argument of `\DeclareFontSeriesDefault`),
- what to do if we are in this context (true case) and
- what to do if we are not (false case).

This allows you to define commands like `\IfBold`, e.g.,

```
\newcommand\IfBold[2]{\IfSeriesContextTF{bf}{#1}{#2}}
```

and then do

```
This is \IfBold{bold}{non-bold} text.
```

and get the appropriate result.

```
496 </2ekernel>
497 <*2ekernel | latexrelease>
498 <latexrelease>\IncludeInRelease{2020/10/01}%
499 <latexrelease> {\IfFontSeriesContextTF}{Font series context}%
500 \DeclareRobustCommand\IfFontSeriesContextTF[1]{%
501 \expand@font@defaults
```

In the beginning we haven't found the context we are looking for.

```
502 \@font@series@contextfalse
```

We store the requested context away for use in the tests.

```
503 \def\requested@test@context{\#1}%
```

The next definition is there to ensure that get a final match during testing even if the current family is non of the meta families (`rm`, `sf` or `tt`). This will then basically tests if the current font family matches the overall default.

```
504 \expandafter\edef\csname ??def@ult\endcsname{\f@family}%
```

Then we run through the meta family list (currently containing just the three values) followed by the artificial meta family `??` and test each of them in turn using `\test@font@series@context` as the testing command.

```
505 \let\@elt\test@font@series@context
506 \@meta@family@list
507 \@elt{??}%
508 \let\@elt\relax
```

Following that we evaluate the status of `\if@font@series@context` to determine which of the remaining arguments (true/false case) we have to execute.

```
509 \if@font@series@context
510 \expandafter\@firstoftwo
511 \else
512 \expandafter\@secondoftwo
513 \fi
514 }
```

(End definition for `\IfFontSeriesContextTF`.)

`\test@font@series@context` This tests the context (stored in `\requested@test@context`) and updates the boolean if the right context is found.

```
515 \def\test@font@series@context#1{%
```

First task is to figure out whether the current family matches `\rmfamily`, `\sffamily`, etc. so in `\reserved@a` we store the value of `\rmdef@ult` (or whatever the given meta family is) and compare that to `\f@family`.

```
516 \edef\reserved@a{\csname #1def@ult\endcsname}%
517 \ifx\f@family\reserved@a
```

If they match we have found the right meta family so we don't need to test any of the remaining meta family and therefore change `\@elt` to `\@gobble`.

```
518 \let\@elt\@gobble
```

Now we have to test if `\f@series` matches the requested context (e.g., whether `\bfseries@rm` has that value if the current meta family is `rm` and we are looking for the `bf` context).

```
519 \expandafter\ifx
520 \csname\requested@test@context series@\#1\endcsname\f@series
```

If yes we change the boolean and are done.

```
521 \font@series@contexttrue
```

If not then maybe the reason is that nothing special was set up for that meta family so we also check now if `\f@series` matches the overall default (e.g., `\bfdef@ult` if we are looking for the bold context). If that matches we change the boolean.

```
522 \else
523 \expandafter\ifx
524 \csname\requested@test@context def@ult\endcsname\f@series
525 \font@series@contexttrue
526 \fi\fi\fi
527 }
```

*(End definition for `\test@font@series@context`.)*

`\if@font@series@context` The boolean to signal if we found the requested font series context.

```
528 \newif\if@font@series@context
```

*(End definition for `\if@font@series@context`.)*

```
529 </2ekernel | latexrelease>
530 <latexrelease>\EndIncludeInRelease
531 <latexrelease>\IncludeInRelease{0000/00/00}%
532 <latexrelease> {\IfFontSeriesContextTF}{Font series context}%
533 <latexrelease>
534 <latexrelease>\let\IfFontSeriesContextTF\@undefined
535 <latexrelease>\let\test@font@series@context\@undefined
536 <latexrelease>\let\if@font@series@context\@undefined
537 <latexrelease>\let\@font@series@contexttrue\@undefined
538 <latexrelease>\let\@font@series@contextfalse\@undefined
539 <latexrelease>\EndIncludeInRelease
540 <*2ekernel>
```

### 3 Supporting nested emphasis

By default L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub>  supports two levels of nested emphasis: if the current font has an upright shape then it switches to `\itshape` otherwise to `\emnnershape` (which defaults to `\upshape`). This means nested emphasis will oscillate between italic and upright shapes.

Sometimes it would be nice to allow for a more lengthy sequence, but instead of providing a fixed one L<sup>A</sup>T<sub>E</sub>X now offers a general mechanism that allows to define arbitrary sequences.

```
\DeclareEmphSequence
\emforce
```

This declaration expects a comma separated list of (font) change declarations corresponding to increasing levels of emphasis. The mechanism tries to be “smart” and verifies that the declarations actually alter the font. If not it will ignore this level and tries the next one—the assumption being that there was a manual font change in the document

to the font that is now supposed to be used for emphasis. Of course, this only works if the declarations in the list actually change the font and not, say, just the color. In such a case one has to use `\emforce` to which directs the mechanism to use the level even if the font attributes haven't changed.

`\emreset` If the nesting is so deep, that the specified levels are exhausted then `\emreset` is used as a final set of declarations (which by default returns back to the upright shape). Any additional nesting levels will then reuse the list from its beginning.

`\DeclareEmphSequence` `\DeclareEmphSequence` expects aclist of declaration. Spaces in the argument are dropped to avoid spurious spaces in the output. The declarations are additive. At the very end the shape is reset using `\emreset` and `\emforce` so that this case is never skipped.<sup>29</sup> Further nested calls restart at the beginning.

```

541 </2ekernel>
542 <*2ekernel | latexrelease>
543 <latexrelease> \IncludeInRelease{2020/02/02}%
544 <latexrelease> {\DeclareEmphSequence}{Nested emph}%
545 \def\DeclareEmphSequence#1{%
546 \protected@edef\emfontdeclare@clist{\zap@space#1, \empty\emforce\emreset}%
547 }

```

By default the it is empty, in which case `\eminnershape` is used by L<sup>A</sup>T<sub>E</sub>X.

```
548 \let\emfontdeclare@clist\empty
```

(End definition for `\DeclareEmphSequence`.)

`\emrest` Reset the font to upright and upper/lower case. With the default rules using `\shapedefault` does that for us but to be on the safe side we do it like this:

```
549 \DeclareRobustCommand\emrest{\upshape\ulcshape}
```

(End definition for `\emrest`.)

`\em` The new definition for `\em` (and implicitly `\emph`) is the same as before as long as `\emfontdeclare@clist` is empty.

```

550 \DeclareRobustCommand\em{%
551 \nomath\em
552 \ifx\emfontdeclare@clist\empty
553 \ifdim \fontdimen1ne\font >\z@
554 \eminnershape \else \itshape \fi
555 \else

```

But if not we use the list to decide how to do emphasis.

We use the current font to check if the declarations have any effect, so even a size change is allowed and identified as a modification (but a color change, for example, isn't). So first we save the current status.

```
556 \edef\em@currfont{\csname curr@fontshape/\f@size\endcsname}%
```

Then we grab the next element from the list and check if it can be used.

```

557 \expandafter\do@emfont@update\emfontdeclare@clist\do@emfont@update
558 \fi
559 }
```

---

<sup>29</sup> Maybe we should not add `\emforce` but allow that case to be skipped as well. Of course, that might result in an endless loop if somebody defines a sequence without any font change and without `\emforce` but ...

```
560 \def\eminnershape{\upshape}
```

(End definition for \em.)

\do@emfont@update We know that the list (if not empty) has at least 2 elements separated by a comma, so we pick up the first in #1 and the rest in #2.

```
561 \def\do@emfont@update#1,#2\do@emfont@update{%
```

First action is to alter the list and move the first entry to the end

```
562 \def\emfontdeclare@clist{-#2,#1}%
```

Then we execute current declaration. Appending \selectfont means one can write just \fontshape{it}{} and that works then too.

```
563 % \typeout{Use: \detokenize{#1}}%
```

```
564 #1\selectfont
```

We then compare the current font with our saved version, but with a slight twist: we add \em@force at the end of the name. Normally this is empty so has no effect but if there was an \emforce as part of #1 it will append a / to the font name (making it invalid) thus this will then always fail the test.

If the test fails we are done and the declarations will be used. Otherwise we will try the next declaration in the sequence.

```
565 \expandafter\ifx\csname \curr@fontshape/\f@size\em@force
```

For the comparison with \ifx we have to expand \em@currfont once as the relevant info is inside.

```
566 \expandafter\endcsname
567 \em@currfont
```

```
568 \expandafter\do@emfont@update\emfontdeclare@clist\do@emfont@update
```

If \emforce was used, we have to undo its effect:

```
569 \else
570 \let\em@force\@empty
571 \fi
572 }
```

(End definition for \do@emfont@update.)

\emforce The definition of \emforce is simple: change \em@force to make the above test always invalid.

```
573 \protected\def\emforce{\def\em@force{/}}
```

```
574 \let\em@force\@empty
```

```
575 </2ekernel | latexrelease>
```

```
576 <latexrelease>\EndIncludeInRelease
```

(End definition for \emforce and \em@force.)

\em \eminnershape These are the older definitions for \em, prior to 2020.

We also have to define the *emphasize* font change command (i.e. \em). This command will look is the current font is sloped (i.e. has a positive \fontdimen1) and will then select either \upshape or \itshape.

```
577 <latexrelease>\IncludeInRelease{2015/01/01}{\DeclareEmphSequence}{Nested emph}%
```

```
578 <latexrelease>\let\DeclareEmphSequence\@undefined
```

```
579 <latexrelease>\let\emfontdeclare@clist\@undefined
```

```
580 <latexrelease>\let\emreset\@undefined
```

```
581 <latexrelease>\let\do@emfont@update\@undefined
```

```

582 〈\latexrelease〉\let\emforce\@undefined
583 〈\latexrelease〉\let\em@force\@undefined
584 〈\latexrelease〉
585 〈\latexrelease〉\DeclareRobustCommand\em
586 〈\latexrelease〉 {＼@nomath\em \ifdim \fontdimen\@ne\font >\z@
587 〈\latexrelease〉 \eminnershape \else \itshape \fi}%
588 〈\latexrelease〉\EndIncludeInRelease
589 〈\latexrelease〉
590 〈\latexrelease〉\IncludeInRelease{0000/00/00}{\DeclareEmphSequence}{Nested emph}%
591 〈\latexrelease〉\DeclareRobustCommand\em
592 〈\latexrelease〉 {＼@nomath\em \ifdim \fontdimen\@ne\font >\z@
593 〈\latexrelease〉 \upshape \else \itshape \fi}%
594 〈\latexrelease〉\let\eminnershape\@undefined
595 〈\latexrelease〉\EndIncludeInRelease
596 〈*2ekernel〉

```

(End definition for `\em` and `\eminnershape`.)

`\not@math@alphabet` This function generates an error message when it is called in math mode. The same function should be defined in `newfont.sty`.

```

597 \def\not@math@alphabet#1#2{%
598 \relax
599 \ifmmode
600 \@latex@error{Command \noexpand#1 invalid in math mode}%
601 {%
602 Please
603 \ifx#2\relax
604 define a new math alphabet^{#1}%
605 if you want to use a special font in math mode%
606 \else

```

We have to a `\noexpand` below to prevent expansion of #2. In case of #1 we can omit this (due to the current definition of robust commands since they do come out right there :-).

```

607 use the math alphabet \noexpand#2 instead of
608 the #1 command%
609 \fi
610 .
611 }%
612 \fi}

```

(End definition for `\not@math@alphabet`.)

Finally we provide two abbreviations to switch to the L<sup>A</sup>T<sub>E</sub>X versions.

```

613 \DeclareRobustCommand\boldmath{\@nomath\boldmath
614 \mathversion{bold}}
615 \DeclareRobustCommand\unboldmath{\@nomath\unboldmath
616 \mathversion{normal}}

```

Here we switch to the default math version by defining the internal macro `\math@version`. We dare not to call `\mathversion` at this place because this would call `\gls@settings`.

```
617 \def\math@version{normal}
```

### 3.1 Legacy

We start by defining a few macros that are part of standard L<sup>A</sup>T<sub>E</sub>X's user interface. The use of these functions is not encouraged, but they will allow to process older documents without changes to the source.

```
\newfont
618 \def\newfont#1#2{\@ifdefinable#1{\font#1=#2\relax}}
(End definition for \newfont.)

\symbol
619 </2ekernel>
620 <*2ekernel | latexrelease>
621 <latexrelease>\IncludeInRelease{2020/10/01}%
622 <latexrelease> {\symbol}{XeTeX change for math}%
623 \ifdefined\XeTeXversion
624 \DeclareRobustCommand\symbol[1]{\Ucharcat#1 12\relax}
625 \else
626 \DeclareRobustCommand\symbol[1]{\char#1\relax}
627 \fi
628 </2ekernel | latexrelease>
629 <| latexrelease>\EndIncludeInRelease
630 <| latexrelease>\IncludeInRelease{0000/00/00}%
631 <| latexrelease> {\symbol}{XeTeX change for math}%
632 <| latexrelease>
633 <| latexrelease>\DeclareRobustCommand\symbol[1]{\char#1\relax}
634 <| latexrelease>
635 <| latexrelease>\EndIncludeInRelease
636 <*2ekernel>
(End definition for \symbol.)
```

### 3.2 Miscellaneous

\@setfontsize This abbreviation is used by L<sup>A</sup>T<sub>E</sub>X's user level size changing commands, such as \large.  
\@setsizesize  
637 \def\@setfontsize#1#2#3{\@nomath#1%  
For the benefit of people relying on keeping the name of the current font command saved in \@currsize we define it. To ensure that \@setfontsize keeps being robust we omit this assignment during times where \protect differs from \typeset@protect.

```
638 \ifx\protect\@typeset@protect
639 \let\@currsize#1%
640 \fi
641 \fontsize{#2}{#3}\selectfont}
```

For compatibility we also define \@setsizesize the 209 command

```
642 <*compat>
643 \def\@setsizesize#1#2#3#4{\@setfontsize#1{#4}{#2}}
644 </compat>
```

*(End definition for \@setfontsize and \@setsizesize.)*

\hexnumber@ To set up L<sup>A</sup>T<sub>E</sub>X's special math character definitions we first provide a macro to generate hexadecimal numbers. It is a rather simple \ifcase.

```
645 \def\hexnumber@#1{\ifcase\number#1
646 0\or 1\or 2\or 3\or 4\or 5\or 6\or 7\or 8\or
647 9\or A\or B\or C\or D\or E\or F\fi}
```

(End definition for \hexnumber@.)

\nfss@text In its simplest form \nfss@text is an \mbox. This will produce unbreakable text outside math and inside math you will get text with the same fonts as outside. The only drawback is that such item won't change sizes in subscripts. But this behavior can be easily changed. With the **amstex** style option one will get a sub style called **amstext** which will redefine the \nfss@text macro to produce correct text in all sizes.

We have to use \def instead of the shorter \let since \mbox is undefined when we reach this point.

```
648 \def\nfss@text#1{{\mbox{#1}}}
```

(End definition for \nfss@text.)

\copyright The definition of \copyright was changed so that it works in other type styles, and to make it robust. We leave the family untouched so that the copyright notice will come out differently if a different font family is in use. This command is commented out, since it is now defined in ltoutenc.dtx.

```
649 %\DeclareRobustCommand\copyright
650 % {\oalign{\hfil
651 % \raise.07ex\hbox{\mdseries\upshape c}\hfil\crcr
652 % \mathhexbox20D}}
```

(End definition for \copyright.)

\normalfont \reset@font The macro \reset@font is used in L<sup>A</sup>T<sub>E</sub>X to switch to a standard font, in order to initialize the current font in situations where typesetting is done in a new visual context (e.g. in a footnote). We define it here to allow the test for the new L<sup>A</sup>T<sub>E</sub>X version above but nevertheless are able to run all kind of mixtures.

The user interface name for \reset@font is \normalfont:

```
653 /2ekernel
654 (*2ekernel | latexrelease)
655 <latexrelease>\IncludeInRelease{2021/06/01}%
656 <latexrelease> {\normalfont}{Add hook to \normalfont}%
657 \DeclareRobustCommand\normalfont{%
```

Instead of calling \usefont, as it was done in the past, we inline the code from \usefont as we want to add the hook before \selectfont, but after all the font attributes are set.

```
658 \fontencoding\encodingdefault
659 \edef\f@family{\familydefault}%
660 \edef\f@series{\seriesdefault}%
661 \edef\f@shape{\shapedefault}%
```

Any earlier \fontseries, etc. should be canceled and we should switch unconditionally to the requested font face so we drop any code that may have been stored in \delayed@f@adjustment.

```
662 \let\delayed@f@adjustment\empty
663 \UseHook{normalfont}%
```

This is the old name for the hook introduced in 2020/02/02. It will be removed in one of the future releases!

```
664 \@defaultfamilyhook % hookname from 2020/02 will vanish
665 \selectfont
666 \let\reset@font\normalfont
(End definition for \normalfont and \reset@font.)
667 % \changes{v3.2g}{2021/03/18}
668 % {Add missing 2020/02/02 latexrelease entry.}
669 </2ekernel | latexrelease>
670 <latexrelease>\EndIncludeInRelease
671 <latexrelease>
672 <latexrelease>\IncludeInRelease{2020/10/01}%
673 <latexrelease> {\normalfont}{Add hook to \normalfont}%
674 <latexrelease>
675 <latexrelease>\DeclareRobustCommand\normalfont{%
676 <latexrelease> \fontencoding\encodingdefault
677 <latexrelease> \edef\f@family{\familydefault}%
678 <latexrelease> \edef\f@series{\seriesdefault}%
679 <latexrelease> \edef\f@shape{\shapedefault}%
680 <latexrelease> \UseHook{normalfont}%
681 <latexrelease> \@defaultfamilyhook % hookname from 2020/02 will vanish
682 <latexrelease> \selectfont
683 <latexrelease>
684 <latexrelease>\let\reset@font\normalfont
685 <latexrelease>
686 <latexrelease>\EndIncludeInRelease
687 <latexrelease>
688 <latexrelease>\IncludeInRelease{2020/02/02}%
689 <latexrelease> {\normalfont}{Add hook to \normalfont}%
690 <latexrelease>
691 <latexrelease>\DeclareRobustCommand\normalfont{%
692 <latexrelease> \fontencoding\encodingdefault
693 <latexrelease> \edef\f@family{\familydefault}%
694 <latexrelease> \edef\f@series{\seriesdefault}%
695 <latexrelease> \edef\f@shape{\shapedefault}%
696 <latexrelease> \@defaultfamilyhook
697 <latexrelease> \selectfont
698 <latexrelease>
699 <latexrelease>\let\reset@font\normalfont
700 <latexrelease>
701 <latexrelease>\let@\defaultfamilyhook@empty
702 <latexrelease>
703 <latexrelease>\EndIncludeInRelease
704 <latexrelease>
705 <latexrelease>\IncludeInRelease{0000/00/00}%
706 <latexrelease> {\normalfont}{Add hook to \normalfont}%
707 <latexrelease>
708 <latexrelease>\DeclareRobustCommand\normalfont
709 <latexrelease> {\usefont\encodingdefault
710 <latexrelease> \familydefault
711 <latexrelease> \seriesdefault
712 <latexrelease> \shapedefault
```

```

713 \relax}
714 \let\reset@font\normalfont
715 \let\@defaultfamilyhook\undefined
716 \let\EndIncludeInRelease
717 \EndIncludeInRelease
718 \EndIncludeInRelease
719 {*2ekernel}

```

We left out the special L<sup>A</sup>T<sub>E</sub>X fonts which are not automatically included in the base version of the font selection since these fonts contain only a few characters which are also included in the AMS fonts so anybody who is using these fonts doesn't need them. But for compatibility reasons we will define these symbols.

```

720 \def\not@base#1{\@latex@error
721 {Command \noexpand#1 not provided in base LATEX2e}%
722 {Load the latexsym or the amsfonts package to
723 define this symbol}}
724 \def\mho{\not@base\mho}
725 \def\Join{\not@base\Join}
726 \def\Box{\not@base\Box}
727 \def\Diamond{\not@base\Diamond}
728 \def\leadsto{\not@base\leadsto}
729 \def\sqsubset{\not@base\sqsubset}
730 \def\sqsupset{\not@base\sqsupset}
731 \def\lhd{\not@base\lhd}
732 \def\unlhd{\not@base\unlhd}
733 \def\rhd{\not@base\rhd}
734 \def\unrhd{\not@base\unrhd}

```

We now initialize all variables set by `\DeclareErrorFont`. These values are not really important since they will be overwritten later on by the definition in `fontdef.ltx`.

However, if `fontdef.cfg` is corrupted then at least a hopefully suitable error font is present.

```

735 \DeclareErrorFont{OT1}{cmr}{m}{n}{10} %% don't modify this setting
736 %% overwrite it in fontdef.cfg
737 %% if necessary

```

We also set some default values for `\f@family` etc. Note that we don't yet have any encodings that comes later. In the past this was implicitly done by `\DeclareErrorFont`.

```

738 \fontfamily{cmr}

```

Previously the default values for series and shape were set by calling `\fontseries` and `\fontshape`, but their action is now delayed until `\selectfont` which isn't called inside the format (to avoid unnecessarily loading a font that may never get used). We therefore have to set `\f@series` and `\f@shape` directly instead.

```

739 \def\f@series{m} % \fontseries{m}
740 \def\f@shape{n} % \fontshape{n}
741 \fontsize{10}{10}

```

The initial `fontenc` package load list. This will get overwritten in `fonttext` and is only provided in case an old `fonttext.cfg` does not define the command:

```

742 \def@\fontenc@load@list{\@elt{T1,OT1}}

```

We now load the customizable parts of NFSS.

```
743 \InputIfFileExists{fonttext.cfg}
744 {\typeout{=====
745 ^^^J%
746 Local config file fonttext.cfg used^ ^^J%
747 ^^^J%
748 =====}%
749 \def\@addtolist##1{\xdef\@filelist{\@filelist,\##1}}%
750 }
751 {\input{fonttext.ltx}}
752 \let\@addtolist\@gobble
Ditto for math although I don't think that we will get a lot of customisation :-)
753 \InputIfFileExists{fontmath.cfg}
754 {\typeout{=====
755 ^^^J%
756 Local config file fontmath.cfg used^ ^^J%
757 ^^^J%
758 =====}%
759 \def\@addtolist##1{\xdef\@filelist{\@filelist,\##1}}%
760 }
761 {\input{fontmath.ltx}}
762 \let\@addtolist\@gobble
```

Then we preload several fonts. This file might be customized *without* changing the behavior of the format (i.e. necessary font definitions will be loaded at runtime if they are not preloaded). This is done in the file `preload.ltx`.

```
763 \InputIfFileExists{preload.cfg}
764 {\typeout{=====
765 ^^^J%
766 Local config file preload.cfg used^ ^^J%
767 ^^^J%
768 =====}%
769 \def\@addtolist##1{\xdef\@filelist{\@filelist,\##1}}%
770 }
771 {\input{preload.ltx}}
772 \let\@addtolist\@gobble
```

`\seriesdefault` After `\seriesdefault` got defined inside `fonttext.ltx` or a `.cfg` file overwriting it, we alter its value by appending `\empty` to it. This will vanish if expanded but allows us to check if the default gets altered (even to the same value) in the document preamble. All we have to do is to save the current value somewhere and later compare the two. For this we use `\seriesdefault@kernel`.

```
773 \expandafter\def\expandafter\seriesdefault\expandafter{\seriesdefault\empty}
774 \let\seriesdefault@kernel\seriesdefault
```

(End definition for `\seriesdefault` and `\seriesdefault@kernel`.)

`\@acci` We also save the values of some accents in `\@acci`, `\@accii` and `\@acciii` so they can be restored by a `minipage` inside a `tabbing` environment.

```
775 \let\@acci\` \let\@accii\` \let\@acciii\=
```

(End definition for `\@acci`, `\@accii`, and `\@acciii`.)

`\cal` Here were the two old *<alphabet identifiers>*.

`\mit`

(End definition for `\cal` and `\mit`.)

776 `\langle /2ekernel \rangle`

# File B

## fontdef.dtx

<-latexrelease> [2021/01/15 v3.0i LaTeX Kernel (<-latexrelease> font setup)]

### 1 Introduction

This file is used to generate the files `fonttext.ltx` (text font declarations) and `fontmath.ltx` (math font declarations), which are used during the format generation. It contains the declaration of the standard text encodings used at the site as well as a minimal subset of font shape groups that NFSS will look at to ensure that the specified encodings are valid.

The math part contains the setup for math encodings as well as the default math symbol declarations that belong to the encoding.

It is possible to change this setup (by using other fonts, or defaults) without losing the ability to process documents written at other sites. Portability in this sense means that a document will compile without errors. It does not mean, however, that identical output will be produced. For this it is necessary that the distributed setup is used at both installations.

### 2 Customization

You are not allowed to change this source file! If you want to change the default encodings and/or the font shape groups preloaded you should create a copy of `fonttext.ltx` under the name `fonttext.cfg` and change this copy. If L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub>  finds a file of this name it will use it, otherwise it uses the standard file which is `fontdef.ltx`.

If you don't plan to use Computer Modern much or at all, it might (!) be a good idea to make your own `fonttext.cfg`. Look at the comments below (docstrip module 'text') to see what should go into such a file.

To change the math font setup use a copy of `fontmath.ltx` under the name `fontmath.cfg` and change this copy. However, dealing with this interface is even more a job for an expert than changing the text font setup — in short, we don't encourage either.

**Warning:** please note that we don't support customised L<sup>A</sup>T<sub>E</sub>X versions. Thus, before sending in a bug report please try your test file with a L<sup>A</sup>T<sub>E</sub>X format which is not customised and send in the log from that version (unless the problem goes away).

Please note: the following standard encodings have to be defined in all local variants of `font....cfg` to guarantee that all L<sup>A</sup>T<sub>E</sub>X installations behave in the same way.

|     |                                                      |
|-----|------------------------------------------------------|
| T1  | Cork T <sub>E</sub> X text encoding                  |
| OT1 | old T <sub>E</sub> X text encoding                   |
| U   | unknown encoding                                     |
| OML | old T <sub>E</sub> X math letters encoding           |
| OMS | old T <sub>E</sub> X math symbols encoding           |
| OMX | old T <sub>E</sub> X math extension symbols encoding |
| TU  | Unicode                                              |

Notice that some of these encodings are ‘old’ in the sense that we hope that they will be superseded soon by encoding standards defined by the  $\text{\TeX}$  user community. Therefore this set of default encodings may change in the future.

The first candidate is OT1 which will soon be replaced by T1, the official  $\text{\TeX}$  text encoding.

**Warning:** If you add additional encodings to this file there is no guarantee any longer that files processable at your installation will also be processable at other installations. Thus, if you make use of such an encoding in your document, e.g. if you intend to typeset in Cyrillic (OT2 encoding), you need to specify this encoding in the preamble of your document prior to sending it to another installation. Once the encoding is specified in that place in your document, the document is processable at all  $\text{\LaTeX}$  installations (provided they have suitable fonts installed).

For this reason we suggest that you define a short package file that sets up an additional encoding used at your site (rather than putting the encoding into this file) since this package can easily be shipped with your document.

### 3 The `docstrip` modules

The following modules are used to direct `docstrip` in generating external files:

|         |                                                |
|---------|------------------------------------------------|
| driver  | produce a documentation driver file            |
| text    | produce the file <code>fonttext.ltx</code>     |
| math    | produce the file <code>fontmath.ltx</code>     |
| cfgtext | produce a dummy <code>fonttext.cfg</code> file |
| cfgmath | produce a dummy <code>fontmath.cfg</code> file |

A typical `docstrip` command file would then have entries like:

```
generateFile{fonttext.ltx}{t}{\from{fontdef.dtx}{text}}
```

### 4 A driver for this document

The next bit of code contains the documentation driver file for  $\text{\TeX}$ , i.e. the file that will produce the documentation you are currently reading. It will be extracted from this file by the `DOCSTRIP` program.

```
1 <*driver>
2 \documentclass{ltxdoc}
3 \GetFileInfo{fontdef.dtx}
4 \begin{document}
5 \DocInput{fontdef.dtx}
6 \end{document}
7
```

### 5 The `fonttext.ltx` file

The identification is done earlier on with a `\ProvidesFile` declaration.

```
8 <*text>
9 \typeout{== Don't modify this file, use a .cfg file instead ==^J}
```

## 5.1 Encodings

This file declares the standard encodings for text and math fonts. All others should be declared in packages or in the documents directly.

For every text encoding there are normally a number of encoding specific commands, e.g. accents, special characters, etc. (The definition for such a command might have to change when the encoding is changed, because the character is in a different position, or not available at all, or the accent is produced in a different way.) This is handled by a general mechanism which is described in `loutenc.dtx`.

By convention, text encoding specific declarations, including the `\DeclareFontEncoding` declaration, are kept in separate file of the form `<enc>enc.def`, e.g. `ot1enc.def`. This allows other applications to make use of the declarations as well.

Similar to the default encoding, the loading of the encoding files for the two major text encodings shouldn't be changed. In particular, the `inputenc` package depends on this.

```
10 \input {omlenc.def}
11 \input {omsenc.def}
```

Documents containing a lot of accented characters should really be using T1 fonts. We therefore load this last so that T1 encoding specific commands are executed as fast as possible (encoding files are no longer reloaded in `fontenc`).

```
12 \input {ot1enc.def}
13 \input {t1enc.def}
14 \input{ts1enc.def}
15 \ifx\Umathcode\@undefined
```

We then set the default text font encoding. This will hopefully change some day to T1. This setting should *not* be changed to produce a portable format.

```
16 \fontencoding{OT1}
```

The initial `fontenc` package load list if an 8-bit TeX engine is used:

```
17 \def\@fontenc@load@list{\@elt{T1,OT1}}
18 \def\rmsubstdefault{cmr}
19 \def\sfsubstdefault{cmss}
20 \def\ttsubstdefault{cmtt}
21 \LoadFontDefinitionFile{TS1}{cmr}
22 \else
```

Unicode.

```
23 \input {tuenc.def}
24 \fontencoding{TU}
```

The initial `fontenc` package load list if a Unicode engine is used:

```
25 \def\@fontenc@load@list{\@elt{TU}}
26 \DeclareFontSubstitution{TU}{lmr}{m}{n}
27 \LoadFontDefinitionFile{TU}{lmr}
28 \LoadFontDefinitionFile{TU}{lmss}
29 \LoadFontDefinitionFile{TU}{lmtt}
30 \def\rmsubstdefault{lmr}
31 \def\sfsubstdefault{lmss}
32 \def\ttsubstdefault{lmtt}
33 \LoadFontDefinitionFile{TS1}{lmr}
```

```
34 \DeclareFontSubstitution{TU}{lmr}{m}{n}
```

End of Unicode branch.

```
35 \fi
```

If different encodings for text fonts are in use one could put the common setup into `\DeclareFontEncodingDefaults`. There is now a better mechanism so using this interface is discouraged!

```
36 \DeclareFontEncodingDefaults{}{}
```

Then we define the default substitution for every encoding. This release of L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub>  assumes that the ec fonts are available. It is possible to change this to point to some other font family (e.g., Times with the appropriate encoding if it is available) without making documents non-portable. However, in such a case documents will produce different page breaks at other sites. The substitution defaults can all be changed without losing portability as long as there are font shape definitions for the selected substitutions.

```
37 \DeclareFontSubstitution{T1}{cmr}{m}{n}
```

```
38 \DeclareFontSubstitution{OT1}{cmr}{m}{n}
```

For every encoding declaration, L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub>  will try to verify that the given substitution information makes sense, i.e. that it is impossible to go into an endless loop if font substitution happens. This is done at the moment the `\begin{document}` is encountered. L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub>  will then check that for every encoding the substitution defaults form a valid font shape group, which means that it will check if there is a `\DeclareFontShape` declaration for this combination. We will therefore load the corresponding .fd files now. If we don't do this they would be loaded at verification time (i.e. at `\begin{document}`) which would delay processing unnecessarily.

**Warning:** Please note that this means that you have to regenerate the format whenever you change any of these .fd files since L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub>  will not read .fd files if it already knows about the encoding/family combination.

The `\nfss@catcodes` ensures that white space is ignored in any definitions made in the fd files.

```
39 \begingroup
40 \nfss@catcodes
41 \input {t1cmr.fd}
42 \input {ot1cmr.fd}
43 \endgroup
```

We also load some other font definition files which are normally needed in a document. This is only done for processing speed and you can comment the next two lines out to save some memory. If necessary these files are then loaded when your document is processed. (Loading .fd files is a less drastic step compared to preloading fonts because the number of fonts is limited 255 at (nearly) every T<sub>E</sub>X installation, while the amount of main memory is not a limiting factor at most installations.)

```
44 \begingroup
45 \nfss@catcodes
46 \input {ot1cmss.fd}
47 \input {ot1cmtt.fd}
48 \endgroup
```

Even with all the precautions it is still possible that NFSS will run into problems, for example, when a `.fd` file contains corrupted data. To guard against such cases NFSS has a very low-level fallback font that is installed with the following line.

```
49 \DeclareErrorFont{OT1}{cmr}{m}{n}{10}
```

This means, “if everything else fails use Computer Modern Roman normal shape at 10pt in the old text encoding”. You can change the font used but the encoding should be the same as the one specified with `\fontencoding` above.

## 5.2 Defaults

To allow the use of `\rmfamily`, `\sffamily`, etc. in documents even if non-standard families are used we provide nine macros which hold the name of the corresponding families, series, and so on. This makes it easy to use other font families (like Times Roman, etc.). One simply has to redefine these defaults.

All these hooks have to be defined in this file but you can change their meaning (except for `\encodingdefault`) without making documents non-portable.

|                               |                                                                                                                                       |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| <code>\encodingdefault</code> | The following three definitions set up the meaning for <code>\rmfamily</code> , <code>\sffamily</code> , and <code>\ttfamily</code> . |
| <code>\rmdefault</code>       | <code>\ifx\Umathcode\@undefined</code>                                                                                                |
| <code>\sfdefault</code>       | <code>\newcommand\encodingdefault{OT1}</code>                                                                                         |
| <code>\ttdefault</code>       | <code>\newcommand\rmdefault{cmr}</code>                                                                                               |
|                               | <code>\newcommand\sffamily{cmss}</code>                                                                                               |
|                               | <code>\newcommand\ttdefault{cmtt}</code>                                                                                              |
|                               | <code>\else</code>                                                                                                                    |
|                               | <code>\newcommand\encodingdefault{TU}</code>                                                                                          |
|                               | <code>\newcommand\rmdefault{lmr}</code>                                                                                               |
|                               | <code>\fontfamily{\rmdefault}</code>                                                                                                  |
|                               | <code>\newcommand\sffamily{lmss}</code>                                                                                               |
|                               | <code>\newcommand\ttdefault{lmtt}</code>                                                                                              |
|                               | <code>\fi</code>                                                                                                                      |
|                               | <code>&lt;/text&gt;</code>                                                                                                            |
|                               | <code>\langle latexrelease \rangle \IncludeInRelease{2017/01/01} %</code>                                                             |
|                               | <code>\langle latexrelease \rangle \encodingdefault{TU encoding default} %</code>                                                     |
|                               | <code>\langle latexrelease \rangle \ifx\Umathcode\@undefined</code>                                                                   |
|                               | <code>\langle latexrelease \rangle \renewcommand\encodingdefault{OT1}</code>                                                          |
|                               | <code>\langle latexrelease \rangle \fontencoding{\encodingdefault}</code>                                                             |
|                               | <code>\langle latexrelease \rangle \renewcommand\rmdefault{cmr}</code>                                                                |
|                               | <code>\langle latexrelease \rangle \fontfamily{\rmdefault}</code>                                                                     |
|                               | <code>\langle latexrelease \rangle \renewcommand\sffamily{cmss}</code>                                                                |
|                               | <code>\langle latexrelease \rangle \renewcommand\ttdefault{cmtt}</code>                                                               |
|                               | <code>\langle latexrelease \rangle \else</code>                                                                                       |
|                               | <code>\langle latexrelease \rangle \renewcommand\encodingdefault{TU}</code>                                                           |
|                               | <code>\langle latexrelease \rangle %done in every job \fontencoding{\encodingdefault}</code>                                          |
|                               | <code>\langle latexrelease \rangle \renewcommand\rmdefault{lmr}</code>                                                                |
|                               | <code>\langle latexrelease \rangle \fontfamily{\rmdefault}</code>                                                                     |
|                               | <code>\langle latexrelease \rangle \renewcommand\sffamily{lmss}</code>                                                                |
|                               | <code>\langle latexrelease \rangle \renewcommand\ttdefault{lmtt}</code>                                                               |
|                               | <code>\langle latexrelease \rangle \fi</code>                                                                                         |
|                               | <code>\langle latexrelease \rangle \EndIncludeInRelease</code>                                                                        |
|                               | <code>\langle latexrelease \rangle \IncludeInRelease{0000/00/00} %</code>                                                             |
|                               | <code>\langle latexrelease \rangle \encodingdefault{TU encoding default} %</code>                                                     |

```

83 \let\fontencoding{\OT1}
84 \let\renewcommand\encodingdefault{\OT1}
85 \let\fontencoding{\encodingdefault}
86 \let\renewcommand\rmdefault{cmr}
87 \let\fontfamily{\rmdefault}
88 \let\renewcommand\sffamily{\cmss}
89 \let\renewcommand\ttdefault{\cmtt}
90 \EndIncludeInRelease
91 {*text}

```

(End definition for `\encodingdefault` and others.)

`\bfdefault` Series changing commands are influenced by the following hooks.  
`\mddefault` 92 `\newcommand\bfdefault{b}` % overwritten below (for rollback)  
93 `\newcommand\mddefault{m}` % overwritten below (for rollback)

(End definition for `\bfdefault` and `\mddefault`.)

`\itdefault` Shape changing commands use the following hooks.

`\sldefault` 94 `\newcommand\itdefault{it}`  
`\scdefault` 95 `\newcommand\sldefault{sl}`  
`\updefault` 96 `\newcommand\scdefault{sc}`  
97 `\newcommand\updefault{up}` % overwritten below (for rollback)

(End definition for `\itdefault` and others.)

```

98 {/text}
99 {*text | \textrun}
100 \let\IncludeInRelease{\@empty}
101 \let\updefault{\@empty} % font defaults change}
102 % \begin{macrocode}
103 \renewcommand\updefault{up}

```

We append `\@empty` to the series value so that we can detect if it got changed via `\def` or `\renewcommand` later.

```

104 \renewcommand\bfdefault{b\@empty}
105 \renewcommand\mddefault{m\@empty}
106 \let\bfdefault@previous\bfdefault
107 \let\mddefault@previous\mddefault
108 {/text | \textrun}
109 \let\EndIncludeInRelease{\@empty}
110 \let\IncludeInRelease{\@empty} % font defaults change}
111 \let\updefault{\@empty} % font defaults change}
112 \let\renewcommand\updefault{n}
113 \let\renewcommand\bfdefault{bx}
114 \let\renewcommand\mddefault{m}
115 \let\renewcommand\updefault{up}
116 \let\bfdefault@previous\undefined
117 \let\mddefault@previous\undefined
118 \let\EndIncludeInRelease{\@empty}
119 {*text}

```

\familydefault Finally we have the hooks that describe the behaviour of the `\normalfont` command.  
\seriesdefault To stay portable, the definition of `\encodingdefault` should *not* be changed and should  
\shapedefault match the setting above for `\fontencoding`. All other values can be set according to  
your taste.

```
120 \newcommand\familydefault{\rmdefault}
121 \newcommand\seriesdefault{\mddefault}
```

In previous releases `\shapedefault` pointed to `\updefault` which resolved to `n`, but  
these days that is no longer the case (and up is wrong when you want to do a reset. So  
we now use `n` explicitly.

```
122 \newcommand\shapedefault{n}
```

(End definition for `\familydefault`, `\seriesdefault`, and `\shapedefault`.)

This finishes the low-level setup in `fonttext.ltx`.

```
123 </text>
```

## 6 The fontmath.ltx file

The identification is done earlier on with a `\ProvidesFile` declaration.

```
124 <*math>
125 \typeout{== Don't modify this file, use a .cfg file instead ==^J}
```

### 6.1 The font encodings used

```
126 \DeclareFontEncoding{OML}{}{}
127 \DeclareFontEncoding{OMS}{}{}
128 \DeclareFontEncoding{OMX}{}{}
```

Finally a declaration for U encoding which serves for all fonts that do not fit standard  
encodings. For math this sets up `\noaccents@` providing for AMS-L<sup>A</sup>T<sub>E</sub>X. This macro  
is used therein to handle accented characters if they are not supported by the font. In  
other words, if fonts with U encoding are used in math, all accents (like from `\breve`) are  
obtained from some other font that has them.

```
129 \DeclareFontEncoding{U}{}{\noaccents@}
```

The encodings for math are next:

```
130 \DeclareFontSubstitution{OML}{cmm}{m}{it}
131 \DeclareFontSubstitution{OMS}{cmsy}{m}{n}
132 \DeclareFontSubstitution{OMX}{cmex}{m}{n}
133 \DeclareFontSubstitution{U}{cmr}{m}{n}

134 \begingroup
135 \nfss@catcodes
136 \input {omlcmm.fd}
137 \input {oms cmsy.fd}
138 \input {omx cmex.fd}
139 \input {ucmr.fd}
140 \endgroup
```

#### 6.1.1 Symbolfont and Alphabet declarations

We now define the basic symbol fonts used by L<sup>A</sup>T<sub>E</sub>X. These four symbol fonts must be  
defined by this file.

It is possible to make the symbol fonts point to other external fonts without losing  
the ability to process documents written at other sites, as long as one defines the same

symbol font names with the same encodings, e.g. `operators` with OT1 etc. If other encodings are used documents become non-portable. Such a change should therefore be done in a package file.

```

141 \DeclareSymbolFont{operators} {OT1}{cmr} {m}{n}
142 \DeclareSymbolFont{letters} {OML}{cmm} {m}{it}
143 \DeclareSymbolFont{symbols} {OMS}{cmsy}{m}{n}
144 \DeclareSymbolFont{largesymbols}{OMX}{cmex}{m}{n}

145 \SetSymbolFont{operators}{bold}{OT1}{cmr}{bx}{n}
146 \SetSymbolFont{letters} {bold}{OML}{cmm}{b}{it}
147 \SetSymbolFont{symbols} {bold}{OMS}{cmsy}{b}{n}

```

Below are the seven math alphabets which are defined by NFSS. Again they must be defined by this file. However, as before you can change the fonts used without losing portability, but you should be careful when changing the encoding since that may make documents come out wrong.

```

148 \DeclareSymbolFontAlphabet{\mathrm}{operators}
149 \DeclareSymbolFontAlphabet{\mathrmnormal}{letters}
150 \DeclareSymbolFontAlphabet{\mathcal}{symbols}
151 \DeclareMathAlphabet{\mathbf}{OT1}{cmr}{bx}{n}
152 \DeclareMathAlphabet{\mathsf}{OT1}{cmss}{m}{n}
153 \DeclareMathAlphabet{\mathit}{OT1}{cmr}{m}{it}
154 \DeclareMathAlphabet{\mathtt}{OT1}{cmtt}{m}{n}

```

Given the currently available fonts we cannot bold-en `\mathbf` and `\mathtt` but in principle one could use ‘ultra bold’ or something. The alphabets defined via `\DeclareSymbolFontAlphabet` will change automatically in a new math version if the corresponding symbol font changes.

```

155 \SetMathAlphabet{\mathsf}{bold}{OT1}{cmss}{bx}{n}
156 \SetMathAlphabet{\mathit}{bold}{OT1}{cmr}{bx}{it}

```

## 6.2 Math font sizes

The declarations below declare the text, script and scriptscript size to be used for each text font size.

All occurrences of sizes longer than a single character are replaced with the macro name that holds them, saving a number of tokens (but losing a bit of speed, so this may not stay this way).

```

157 \DeclareMathSizes{5}{5}{5}{5}
158 \DeclareMathSizes{6}{6}{5}{5}
159 \DeclareMathSizes{7}{7}{5}{5}
160 \DeclareMathSizes{8}{8}{6}{5}
161 \DeclareMathSizes{9}{9}{6}{5}
162 \DeclareMathSizes{\@xpt}{\@xpt}{7}{5}
163 \DeclareMathSizes{\@xipt}{\@xipt}{8}{6}
164 \DeclareMathSizes{\@xiipt}{\@xiipt}{8}{6}
165 \DeclareMathSizes{\@xivpt}{\@xivpt}{\@xpt}{7}
166 \DeclareMathSizes{\@xviipt}{\@xviipt}{\@xiipt}{\@xpt}
167 \DeclareMathSizes{\@xxpt}{\@xxpt}{\@xivpt}{\@xiipt}
168 \DeclareMathSizes{\@xxvpt}{\@xxvpt}{\@xxpt}{\@xviipt}

```

## 6.3 The math symbol assignments

We start by setting up math codes for most of the characters typed in directly from the keyboard. Most of them are normally already setup up in the same way by IniT<sub>E</sub>X. However, we repeat them here to have a complete setup which can be exchanged with another if desired.

### 6.3.1 The letters

```
169 \DeclareMathSymbol{a}{\mathalpha}{letters}{`a}
170 \DeclareMathSymbol{b}{\mathalpha}{letters}{`b}
171 \DeclareMathSymbol{c}{\mathalpha}{letters}{`c}
172 \DeclareMathSymbol{d}{\mathalpha}{letters}{`d}
173 \DeclareMathSymbol{e}{\mathalpha}{letters}{`e}
174 \DeclareMathSymbol{f}{\mathalpha}{letters}{`f}
175 \DeclareMathSymbol{g}{\mathalpha}{letters}{`g}
176 \DeclareMathSymbol{h}{\mathalpha}{letters}{`h}
177 \DeclareMathSymbol{i}{\mathalpha}{letters}{`i}
178 \DeclareMathSymbol{j}{\mathalpha}{letters}{`j}
179 \DeclareMathSymbol{k}{\mathalpha}{letters}{`k}
180 \DeclareMathSymbol{l}{\mathalpha}{letters}{`l}
181 \DeclareMathSymbol{m}{\mathalpha}{letters}{`m}
182 \DeclareMathSymbol{n}{\mathalpha}{letters}{`n}
183 \DeclareMathSymbol{o}{\mathalpha}{letters}{`o}
184 \DeclareMathSymbol{p}{\mathalpha}{letters}{`p}
185 \DeclareMathSymbol{q}{\mathalpha}{letters}{`q}
186 \DeclareMathSymbol{r}{\mathalpha}{letters}{`r}
187 \DeclareMathSymbol{s}{\mathalpha}{letters}{`s}
188 \DeclareMathSymbol{t}{\mathalpha}{letters}{`t}
189 \DeclareMathSymbol{u}{\mathalpha}{letters}{`u}
190 \DeclareMathSymbol{v}{\mathalpha}{letters}{`v}
191 \DeclareMathSymbol{w}{\mathalpha}{letters}{`w}
192 \DeclareMathSymbol{x}{\mathalpha}{letters}{`x}
193 \DeclareMathSymbol{y}{\mathalpha}{letters}{`y}
194 \DeclareMathSymbol{z}{\mathalpha}{letters}{`z}

195 \DeclareMathSymbol{A}{\mathalpha}{letters}{`A}
196 \DeclareMathSymbol{B}{\mathalpha}{letters}{`B}
197 \DeclareMathSymbol{C}{\mathalpha}{letters}{`C}
198 \DeclareMathSymbol{D}{\mathalpha}{letters}{`D}
199 \DeclareMathSymbol{E}{\mathalpha}{letters}{`E}
200 \DeclareMathSymbol{F}{\mathalpha}{letters}{`F}
201 \DeclareMathSymbol{G}{\mathalpha}{letters}{`G}
202 \DeclareMathSymbol{H}{\mathalpha}{letters}{`H}
203 \DeclareMathSymbol{I}{\mathalpha}{letters}{`I}
204 \DeclareMathSymbol{J}{\mathalpha}{letters}{`J}
205 \DeclareMathSymbol{K}{\mathalpha}{letters}{`K}
206 \DeclareMathSymbol{L}{\mathalpha}{letters}{`L}
207 \DeclareMathSymbol{M}{\mathalpha}{letters}{`M}
208 \DeclareMathSymbol{N}{\mathalpha}{letters}{`N}
209 \DeclareMathSymbol{O}{\mathalpha}{letters}{`O}
210 \DeclareMathSymbol{P}{\mathalpha}{letters}{`P}
211 \DeclareMathSymbol{Q}{\mathalpha}{letters}{`Q}
212 \DeclareMathSymbol{R}{\mathalpha}{letters}{`R}
213 \DeclareMathSymbol{S}{\mathalpha}{letters}{`S}
```

```

214 \DeclareMathSymbol{T}{\mathalpha}{letters}{`T}
215 \DeclareMathSymbol{U}{\mathalpha}{letters}{`U}
216 \DeclareMathSymbol{V}{\mathalpha}{letters}{`V}
217 \DeclareMathSymbol{W}{\mathalpha}{letters}{`W}
218 \DeclareMathSymbol{X}{\mathalpha}{letters}{`X}
219 \DeclareMathSymbol{Y}{\mathalpha}{letters}{`Y}
220 \DeclareMathSymbol{Z}{\mathalpha}{letters}{`Z}

```

### 6.3.2 The digits

```

221 \DeclareMathSymbol{0}{\mathalpha}{operators}{`0}
222 \DeclareMathSymbol{1}{\mathalpha}{operators}{`1}
223 \DeclareMathSymbol{2}{\mathalpha}{operators}{`2}
224 \DeclareMathSymbol{3}{\mathalpha}{operators}{`3}
225 \DeclareMathSymbol{4}{\mathalpha}{operators}{`4}
226 \DeclareMathSymbol{5}{\mathalpha}{operators}{`5}
227 \DeclareMathSymbol{6}{\mathalpha}{operators}{`6}
228 \DeclareMathSymbol{7}{\mathalpha}{operators}{`7}
229 \DeclareMathSymbol{8}{\mathalpha}{operators}{`8}
230 \DeclareMathSymbol{9}{\mathalpha}{operators}{`9}

```

### 6.3.3 Punctuation, brace, etc. keys

```

231 \DeclareMathSymbol{!}{\mathclose}{operators}{`21}
232 \DeclareMathSymbol{*}{\mathbin}{symbols}{`03} % \ast
233 \DeclareMathSymbol{+}{\mathbin}{operators}{`2B}
234 \DeclareMathSymbol{,}{\mathpunct}{letters}{`3B}
235 \DeclareMathSymbol{-}{\mathbin}{symbols}{`00}
236 \DeclareMathSymbol{.}{\mathord}{letters}{`3A}
237 \DeclareMathSymbol{:}{\mathrel}{operators}{`3A}
238 \DeclareMathSymbol{;}{\mathpunct}{operators}{`3B}
239 \DeclareMathSymbol{=}{\mathrel}{operators}{`3D}
240 \DeclareMathSymbol{?}{\mathclose}{operators}{`3F}

```

The following symbols are defined as delimiters below which automatically defines them as math symbols.

```

241 \% \DeclareMathSymbol{()}{\mathopen}{operators}{`28}
242 \% \DeclareMathSymbol{}{)}{\mathclose}{operators}{`29}
243 \% \DeclareMathSymbol{/}{\mathord}{letters}{`3D}
244 \% \DeclareMathSymbol[]{}{\mathopen}{operators}{`5B}
245 \% \DeclareMathSymbol[]{}{\mathclose}{operators}{`5D}
246 \% \DeclareMathSymbol[]{}{\mathord}{symbols}{`6A}
247 \% \DeclareMathSymbol{<}{\mathrel}{letters}{`3C}
248 \% \DeclareMathSymbol{>}{\mathrel}{letters}{`3E}

```

Should all of the following being activated by default? Probably not.

```

249 \% \DeclareMathSymbol{'{}{\mathopen}{symbols}{`66}
250 \% \DeclareMathSymbol{'{}{\mathclose}{symbols}{`67}
251 \% \DeclareMathSymbol{'\\}{\mathord}{symbols}{`6E} % \backslash
252 \mathcode`\"=8000 % \space
253 \mathcode`'=8000 % ^\prime
254 \mathcode`_="8000 % \

```

### 6.3.4 Delimitercodes for characters

[to be completed]

Finally,  $\text{\rm Init}\text{\rm EX}$  sets all  $\text{\rm \delcode}$  values to -1, except  $\text{\rm \delcode'=.0}$

```

255 \DeclareMathDelimiter{()}{\mathopen}{operators}{28}{largesymbols}{00}
256 \DeclareMathDelimiter{}{\mathclose}{operators}{29}{largesymbols}{01}
257 \DeclareMathDelimiter{[]}{\mathopen}{operators}{5B}{largesymbols}{02}
258 \DeclareMathDelimiter{}{\mathclose}{operators}{5D}{largesymbols}{03}

```

The next two are considered to be relations when not used in the context of a delimiter! And worse, they do even represent different glyphs when being used as delimiter and not as delimiter. This is a user level syntax inherited from plain TeX. Therefore we explicitly redefine the math symbol definitions for these symbols afterwards.

```

259 \DeclareMathDelimiter{<}{\mathopen}{symbols}{68}{largesymbols}{0A}
260 \DeclareMathDelimiter{>}{\mathclose}{symbols}{69}{largesymbols}{0B}
261 \DeclareMathSymbol{<}{\mathrel}{letters}{3C}
262 \DeclareMathSymbol{>}{\mathrel}{letters}{3E}

```

And here is another case where the non-delimiter version produces a glyph different from the delimiter version.

```

263 \DeclareMathDelimiter{/}{\mathord}{operators}{2F}{largesymbols}{0E}
264 \DeclareMathSymbol{/}{\mathord}{letters}{3D}
265 \DeclareMathDelimiter{|}{\mathord}{symbols}{6A}{largesymbols}{0C}
266 \expandafter\DeclareMathDelimiter@\backslashchar
267 \mathord{symbols}{6E}{largesymbols}{0F}

```

N.B. { and } should NOT get delcodes; otherwise parameter grouping fails!

## 6.4 Symbols accessed via control sequences

### 6.4.1 Greek letters

```

268 \DeclareMathSymbol{\alpha}{\mathord}{letters}{0B}
269 \DeclareMathSymbol{\beta}{\mathord}{letters}{0C}
270 \DeclareMathSymbol{\gamma}{\mathord}{letters}{0D}
271 \DeclareMathSymbol{\delta}{\mathord}{letters}{0E}
272 \DeclareMathSymbol{\epsilon}{\mathord}{letters}{0F}
273 \DeclareMathSymbol{\zeta}{\mathord}{letters}{10}
274 \DeclareMathSymbol{\eta}{\mathord}{letters}{11}
275 \DeclareMathSymbol{\theta}{\mathord}{letters}{12}
276 \DeclareMathSymbol{\iota}{\mathord}{letters}{13}
277 \DeclareMathSymbol{\kappa}{\mathord}{letters}{14}
278 \DeclareMathSymbol{\lambda}{\mathord}{letters}{15}
279 \DeclareMathSymbol{\mu}{\mathord}{letters}{16}
280 \DeclareMathSymbol{\nu}{\mathord}{letters}{17}
281 \DeclareMathSymbol{\xi}{\mathord}{letters}{18}
282 \DeclareMathSymbol{\pi}{\mathord}{letters}{19}
283 \DeclareMathSymbol{\rho}{\mathord}{letters}{1A}
284 \DeclareMathSymbol{\sigma}{\mathord}{letters}{1B}
285 \DeclareMathSymbol{\tau}{\mathord}{letters}{1C}
286 \DeclareMathSymbol{\upsilon}{\mathord}{letters}{1D}
287 \DeclareMathSymbol{\phi}{\mathord}{letters}{1E}
288 \DeclareMathSymbol{\chi}{\mathord}{letters}{1F}
289 \DeclareMathSymbol{\psi}{\mathord}{letters}{20}
290 \DeclareMathSymbol{\omega}{\mathord}{letters}{21}
291 \DeclareMathSymbol{\varepsilon}{\mathord}{letters}{22}
292 \DeclareMathSymbol{\vartheta}{\mathord}{letters}{23}
293 \DeclareMathSymbol{\varpi}{\mathord}{letters}{24}

```

```

294 \DeclareMathSymbol{\varrho}{\mathord}{letters}{"25}
295 \DeclareMathSymbol{\varsigma}{\mathord}{letters}{"26}
296 \DeclareMathSymbol{\varphi}{\mathord}{letters}{"27}
297 \DeclareMathSymbol{\Gamma}{\mathalpha}{operators}{"00}
298 \DeclareMathSymbol{\Delta}{\mathalpha}{operators}{"01}
299 \DeclareMathSymbol{\Theta}{\mathalpha}{operators}{"02}
300 \DeclareMathSymbol{\Lambda}{\mathalpha}{operators}{"03}
301 \DeclareMathSymbol{\Xi}{\mathalpha}{operators}{"04}
302 \DeclareMathSymbol{\Pi}{\mathalpha}{operators}{"05}
303 \DeclareMathSymbol{\Sigma}{\mathalpha}{operators}{"06}
304 \DeclareMathSymbol{\Upsilon}{\mathalpha}{operators}{"07}
305 \DeclareMathSymbol{\Phi}{\mathalpha}{operators}{"08}
306 \DeclareMathSymbol{\Psi}{\mathalpha}{operators}{"09}
307 \DeclareMathSymbol{\Omega}{\mathalpha}{operators}{"0A}

```

#### 6.4.2 Ordinary symbols

```

308 \DeclareMathSymbol{\aleph}{\mathord}{symbols}{"40}
309 \DeclareMathSymbol{\imath}{\mathord}{letters}{"7B}
310 \DeclareMathSymbol{\jmath}{\mathord}{letters}{"7C}
311 \DeclareMathSymbol{\ell}{\mathord}{letters}{"60}
312 \DeclareMathSymbol{\wp}{\mathord}{letters}{"7D}
313 \DeclareMathSymbol{\Re}{\mathord}{symbols}{"3C}
314 \DeclareMathSymbol{\Im}{\mathord}{symbols}{"3D}
315 \DeclareMathSymbol{\partial}{\mathord}{letters}{"40}
316 \DeclareMathSymbol{\infty}{\mathord}{symbols}{"31}
317 \DeclareMathSymbol{\prime}{\mathord}{symbols}{"30}
318 \DeclareMathSymbol{\emptyset}{\mathord}{symbols}{"3B}
319 \DeclareMathSymbol{\nabla}{\mathord}{symbols}{"72}
320 \DeclareMathSymbol{\top}{\mathord}{symbols}{"3E}
321 \DeclareMathSymbol{\bot}{\mathord}{symbols}{"3F}
322 \DeclareMathSymbol{\triangle}{\mathord}{symbols}{"34}
323 \DeclareMathSymbol{\forall}{\mathord}{symbols}{"38}
324 \DeclareMathSymbol{\exists}{\mathord}{symbols}{"39}
325 \DeclareMathSymbol{\neg}{\mathord}{symbols}{"3A}

```

Alias:

```

326 % \let\lnot=\neg
327 \DeclareMathSymbol{\lnot}{\mathord}{symbols}{"3A}
328 \DeclareMathSymbol{\flat}{\mathord}{letters}{"5B}
329 \DeclareMathSymbol{\natural}{\mathord}{letters}{"5C}
330 \DeclareMathSymbol{\sharp}{\mathord}{letters}{"5D}
331 \DeclareMathSymbol{\clubsuit}{\mathord}{symbols}{"7C}
332 \DeclareMathSymbol{\diamondsuit}{\mathord}{symbols}{"7D}
333 \DeclareMathSymbol{\heartsuit}{\mathord}{symbols}{"7E}
334 \DeclareMathSymbol{\spadesuit}{\mathord}{symbols}{"7F}

335 \ DeclareRobustCommand{\hbar}{\mathchar'26\mkern-9mu h}
336 \ DeclareRobustCommand{\surd}{\mathchar"1270}
337 \ DeclareRobustCommand{\angle}{\vbox{\ialign{$\m@th\scriptstyle##$\crcr
338 \not\mathrel{\mkern14mu}\crcr
339 \noalign{\nointerlineskip}
340 \mkern2.5mu\leaders\hrule\height.34pt\hfill\mkern2.5mu\crcr}}}}

```

#### 6.4.3 Large Operators

```

341 \DeclareMathSymbol{\coprod}{\mathop}{largesymbols}{"60}

```

```

342 \DeclareMathSymbol{\bigvee}{\mathop}{largesymbols}{57}
343 \DeclareMathSymbol{\bigwedge}{\mathop}{largesymbols}{56}
344 \DeclareMathSymbol{\biguplus}{\mathop}{largesymbols}{55}
345 \DeclareMathSymbol{\bigcap}{\mathop}{largesymbols}{54}
346 \DeclareMathSymbol{\bigcup}{\mathop}{largesymbols}{53}
347 \DeclareMathSymbol{\intop}{\mathop}{largesymbols}{52}
348 \ DeclareRobustCommand\int{\intop\nolimits}
349 \DeclareMathSymbol{\prod}{\mathop}{largesymbols}{51}
350 \DeclareMathSymbol{\sum}{\mathop}{largesymbols}{50}
351 \DeclareMathSymbol{\bigotimes}{\mathop}{largesymbols}{4E}
352 \DeclareMathSymbol{\bigoplus}{\mathop}{largesymbols}{4C}
353 \DeclareMathSymbol{\bigodot}{\mathop}{largesymbols}{4A}
354 \DeclareMathSymbol{\ointop}{\mathop}{largesymbols}{48}
355 \ DeclareRobustCommand\oint{\ointop\nolimits}
356 \DeclareMathSymbol{\bigsqcup}{\mathop}{largesymbols}{46}
357 \DeclareMathSymbol{\smallint}{\mathop}{symbols}{73}

```

#### 6.4.4 Binary symbols

```

358 \DeclareMathSymbol{\triangleleft}{\mathbin}{letters}{2F}
359 \DeclareMathSymbol{\triangleright}{\mathbin}{letters}{2E}
360 \DeclareMathSymbol{\bigtriangleup}{\mathbin}{symbols}{34}
361 \DeclareMathSymbol{\bigtriangledown}{\mathbin}{symbols}{35}

```

Alias:

```

362 % \let \varbigtriangledown \bigtriangledown
363 % \let \varbigtriangleup \bigtriangleup
364 \DeclareMathSymbol{\varbigtriangleup}{\mathbin}{symbols}{34}
365 \DeclareMathSymbol{\varbigtriangledown}{\mathbin}{symbols}{35}

```

These last two synonyms are needed because the *stmaryrd* package redefines them as Operators.

```

366 \DeclareMathSymbol{\wedge}{\mathbin}{symbols}{5E}
367 \DeclareMathSymbol{\vee}{\mathbin}{symbols}{5F}

```

Alias:

```

368 % \let\land=\wedge
369 % \let\lor=\vee
370 \DeclareMathSymbol{\land}{\mathbin}{symbols}{5E}
371 \DeclareMathSymbol{\lor}{\mathbin}{symbols}{5F}
372 \DeclareMathSymbol{\cap}{\mathbin}{symbols}{5C}
373 \DeclareMathSymbol{\cup}{\mathbin}{symbols}{5B}
374 \DeclareMathSymbol{\ddagger}{\mathbin}{symbols}{7A}
375 \DeclareMathSymbol{\dagger}{\mathbin}{symbols}{79}
376 \DeclareMathSymbol{\sqcap}{\mathbin}{symbols}{75}
377 \DeclareMathSymbol{\sqcup}{\mathbin}{symbols}{74}
378 \DeclareMathSymbol{\uplus}{\mathbin}{symbols}{5D}
379 \DeclareMathSymbol{\amalg}{\mathbin}{symbols}{71}
380 \DeclareMathSymbol{\diamond}{\mathbin}{symbols}{05}
381 \DeclareMathSymbol{\bullet}{\mathbin}{symbols}{0F}
382 \DeclareMathSymbol{\wr}{\mathbin}{symbols}{6F}
383 \DeclareMathSymbol{\div}{\mathbin}{symbols}{04}
384 \DeclareMathSymbol{\odot}{\mathbin}{symbols}{0C}
385 \DeclareMathSymbol{\oslash}{\mathbin}{symbols}{0B}
386 \DeclareMathSymbol{\otimes}{\mathbin}{symbols}{0A}
387 \DeclareMathSymbol{\ominus}{\mathbin}{symbols}{09}

```

```

388 \DeclareMathSymbol{\oplus}{\mathbin}{symbols}{08}
389 \DeclareMathSymbol{\mp}{\mathbin}{symbols}{07}
390 \DeclareMathSymbol{\pm}{\mathbin}{symbols}{06}
391 \DeclareMathSymbol{\circ}{\mathbin}{symbols}{0E}
392 \DeclareMathSymbol{\bigcirc}{\mathbin}{symbols}{0D}
393 \DeclareMathSymbol{\setminus}{\mathbin}{symbols}{6E}
394 \DeclareMathSymbol{\cdotp}{\mathbin}{symbols}{01}
395 \DeclareMathSymbol{\ast}{\mathbin}{symbols}{03}
396 \DeclareMathSymbol{\times}{\mathbin}{symbols}{02}
397 \DeclareMathSymbol{\star}{\mathbin}{letters}{3F}

```

#### 6.4.5 Relations

```

398 \DeclareMathSymbol{\propto}{\mathrel}{symbols}{2F}
399 \DeclareMathSymbol{\sqsubseteq}{\mathrel}{symbols}{76}
400 \DeclareMathSymbol{\sqsupseteq}{\mathrel}{symbols}{77}
401 \DeclareMathSymbol{\parallel}{\mathrel}{symbols}{6B}
402 \DeclareMathSymbol{\mid}{\mathrel}{symbols}{6A}
403 \DeclareMathSymbol{\dashv}{\mathrel}{symbols}{61}
404 \DeclareMathSymbol{\vdash}{\mathrel}{symbols}{60}
405 \DeclareMathSymbol{\nearrow}{\mathrel}{symbols}{25}
406 \DeclareMathSymbol{\searrow}{\mathrel}{symbols}{26}
407 \DeclareMathSymbol{\nwarrow}{\mathrel}{symbols}{2D}
408 \DeclareMathSymbol{\swarrow}{\mathrel}{symbols}{2E}
409 \DeclareMathSymbol{\Leftrightarrow}{\mathrel}{symbols}{2C}
410 \DeclareMathSymbol{\Leftarrow}{\mathrel}{symbols}{28}
411 \DeclareMathSymbol{\Rightarrow}{\mathrel}{symbols}{29}
412 \DeclareRobustCommand{\neq}{\not=}

```

As `\neq` is robust we should not use `\let` to define `\ne` as then it would change if `\neq` changes.

```
413 \DeclareRobustCommand{\ne}{\not=}
```

It would ok to use `\let` for those declared by `\DeclareMathSymbol` but for a cleaner interface we avoid it always (just in case the internals change).

```
414 \DeclareMathSymbol{\leq}{\mathrel}{symbols}{14}
415 \DeclareMathSymbol{\geq}{\mathrel}{symbols}{15}
```

Alias:

```

416 % \let\le=\leq
417 % \let\ge=\geq
418 \DeclareMathSymbol{\le}{\mathrel}{symbols}{14}
419 \DeclareMathSymbol{\ge}{\mathrel}{symbols}{15}
420 \DeclareMathSymbol{\succ}{\mathrel}{symbols}{1F}
421 \DeclareMathSymbol{\prec}{\mathrel}{symbols}{1E}
422 \DeclareMathSymbol{\approx}{\mathrel}{symbols}{19}
423 \DeclareMathSymbol{\succeq}{\mathrel}{symbols}{17}
424 \DeclareMathSymbol{\preceq}{\mathrel}{symbols}{16}
425 \DeclareMathSymbol{\supset}{\mathrel}{symbols}{1B}
426 \DeclareMathSymbol{\subset}{\mathrel}{symbols}{1A}
427 \DeclareMathSymbol{\supseteq}{\mathrel}{symbols}{13}
428 \DeclareMathSymbol{\subseteq}{\mathrel}{symbols}{12}
429 \DeclareMathSymbol{\in}{\mathrel}{symbols}{32}
430 \DeclareMathSymbol{\ni}{\mathrel}{symbols}{33}

```

Alias:

```
431 % \let\owns=\ni
```

```

432 \DeclareMathSymbol{\owns}{\mathrel}{symbols}{33}
433 \DeclareMathSymbol{\gg}{\mathrel}{symbols}{1D}
434 \DeclareMathSymbol{\ll}{\mathrel}{symbols}{1C}
435 \DeclareMathSymbol{\not}{\mathrel}{symbols}{36}
436 \DeclareMathSymbol{\leftrightarrow}{\mathrel}{symbols}{24}
437 \DeclareMathSymbol{\leftarrow}{\mathrel}{symbols}{20}
438 \DeclareMathSymbol{\rightarrow}{\mathrel}{symbols}{21}

Alias:
439 % \let\gets=\leftarrow
440 % \let\to=\rightarrow
441 \DeclareMathSymbol{\gets}{\mathrel}{symbols}{20}
442 \DeclareMathSymbol{\to}{\mathrel}{symbols}{21}
443 \DeclareMathSymbol{\mapstochar}{\mathrel}{symbols}{37}
444 \ DeclareRobustCommand{\mapsto}{\mapstochar\rightarrow}
445 \DeclareMathSymbol{\sim}{\mathrel}{symbols}{18}
446 \DeclareMathSymbol{\simeq}{\mathrel}{symbols}{27}
447 \DeclareMathSymbol{\perp}{\mathrel}{symbols}{3F}
448 \DeclareMathSymbol{\equiv}{\mathrel}{symbols}{11}
449 \DeclareMathSymbol{\asymp}{\mathrel}{symbols}{10}
450 \DeclareMathSymbol{\smile}{\mathrel}{letters}{5E}
451 \DeclareMathSymbol{\frown}{\mathrel}{letters}{5F}
452 \DeclareMathSymbol{\leftharpoonup}{\mathrel}{letters}{28}
453 \DeclareMathSymbol{\leftharpoondown}{\mathrel}{letters}{29}
454 \DeclareMathSymbol{\rightharpoonup}{\mathrel}{letters}{2A}
455 \DeclareMathSymbol{\rightharpoondown}{\mathrel}{letters}{2B}

```

Here cometh much profligate robustification of math constructs. Warning: some of these commands may become non-robust if an AMS package is loaded.

Further potential problems: some math font packages may make unfortunate assumptions about some of these definitions that are not true of the robust versions we need.

```

456 \DeclareRobustCommand
457 \cong{\mathrel{\mathpalette\overeq\sim}} % congruence sign
458 \def\overeq#1#2{\lower.5\p@\vbox{\lineskip\maxdimen\lineskip-.5\p@
459 \ialign{$\m@th#1\hfil##\hfil$\crcr#2\crcr=\crcr}}}
460 \DeclareRobustCommand
461 \notin{\mathrel{\mathpalette\cncel\in}}
462 \def\cncel#1#2{\m@th\ooalign{$\hfil#1\mkern1mu/\hfil$\crcr$#1#2$}}
463 \DeclareRobustCommand
464 \rightleftharpoons{\mathrel{\mathpalette\rlh@{}{}}}
465 \def\rlh@#1{\vcenter{\m@th\hbox{\ooalign{\raise2pt
466 \hbox{$\#1\rightharpoonup$}\crcr
467 $\#1\leftharpoondown$}}}}
468 \DeclareRobustCommand
469 \doteq{\mathrel{\textstyle.\over=}}

```

#### 6.4.6 Arrows

```

470 \DeclareRobustCommand
471 \joinrel{\mathrel{\mkern-3mu}}
472 \DeclareRobustCommand
473 \relbar{\mathrel{\smash-}} % \smash, because -
474 % has the same height as +

```

In contrast to `plain.tex` `\Relbar` got braces around the equal sign to guard against it being “math active” expanding to `\futurelet...`. This might be the case when packages are implementing shorthands for math, e.g. `=` meaning `\Rightarrow` etc. It would actually be better not to use `=` in such definitions but instead define something like `\mathequalsign` and use this. However we can’t do this now as it would break other math layouts where characters are in different places (since those wouldn’t know about the need for a new command name).

```

475 \DeclareRobustCommand
476 \Relbar{\mathrel{=}}
477 \DeclareMathSymbol{\lhook}{\mathrel}{letters}{`2C}
478 \DeclareRobustCommand\hookrightarrow{\lhook\joinrel\rightarrow}
479 \DeclareMathSymbol{\rhook}{\mathrel}{letters}{`2D}
480 \DeclareRobustCommand\hookleftarrow{\leftarrow\joinrel\rhook}
481 \DeclareRobustCommand
482 \bowtie{\mathrel\triangleleft\joinrel\mathrel\triangleleft}
483 \DeclareRobustCommand
484 \models{\mathrel{!}\joinrel\Relbar}
485 \DeclareRobustCommand
486 \Longrightarrow{\Relbar\joinrel\rightarrow}

```

LaTeX Change: `\longrightarrow` and `\longleftarrow` redefined to make them robust.

```

487 \DeclareRobustCommand\longrightarrow
488 {\relbar\joinrel\rightarrow}
489 \DeclareRobustCommand\longleftarrow
490 {\leftarrow\joinrel\relbar}
491 \DeclareRobustCommand
492 \Longrightarrow{\Leftarrow\joinrel\Relbar}
493 \DeclareRobustCommand
494 \longmapsto{\mapstochar\longrightarrow}
495 \DeclareRobustCommand
496 \longleftrightarrow{\leftarrow\joinrel\rightarrow}
497 \DeclareRobustCommand
498 \Longleftrightarrow{\Leftarrow\joinrel\rightarrow}
499 \DeclareRobustCommand
500 \iiff{;}{\Longleftrightarrow{};}

```

#### 6.4.7 Punctuation symbols

```

501 \DeclareMathSymbol{\ldotp}{\mathpunct}{letters}{`3A}
502 \DeclareMathSymbol{\cdotp}{\mathpunct}{symbols}{`01}
503 \DeclareMathSymbol{\colon}{\mathpunct}{operators}{`3A}

```

This is commented out, since `\ldots` is now defined in `ltoutenc.dtx`.

```

504 %\def\@ldots{\mathinner{\ldotp\ldotp\ldotp}}
505 %\ DeclareRobustCommand\ldots
506 % {\relax\ifmmode\@ldots\else\mbox{$\m@th\@ldots$}\fi}
507 \DeclareRobustCommand
508 \cdots{\mathinner{\cdotp\cdotp\cdotp}}
509 \DeclareRobustCommand
510 \vdots{\vbox{\baselineskip4\p@\lineskiplimit\z@
511 \kern6\p@\hbox{.}\hbox{.}\hbox{.}}}
512 \DeclareRobustCommand
513 \ddots{\mathinner{\mkern1mu\raise7\p@
514 \vbox{\kern7\p@\hbox{.}\hbox{.}}\mkern2mu}

```

```
515 \raise4\p@\hbox{.}\mkern2mu\raise\p@\hbox{.}\mkern1mu}}
```

#### 6.4.8 Math accents

```
516 \DeclareMathAccent{\acute}{\mathalpha}{operators}{13}
517 \DeclareMathAccent{\grave}{\mathalpha}{operators}{12}
518 \DeclareMathAccent{\ddot}{\mathalpha}{operators}{7F}
519 \DeclareMathAccent{\tilde}{\mathalpha}{operators}{7E}
520 \DeclareMathAccent{\bar}{\mathalpha}{operators}{16}
521 \DeclareMathAccent{\breve}{\mathalpha}{operators}{15}
522 \DeclareMathAccent{\check}{\mathalpha}{operators}{14}
523 \DeclareMathAccent{\hat}{\mathalpha}{operators}{5E}
524 \DeclareMathAccent{\vec}{\mathord}{letters}{7E}
525 \DeclareMathAccent{\dot}{\mathalpha}{operators}{5F}
526 \DeclareMathAccent{\widetilde}{\mathord}{largesymbols}{65}
527 \DeclareMathAccent{\widehat}{\mathord}{largesymbols}{62}
```

For some reason plain TeX never bothered to provide a ring accent in math (although it is available in the fonts), but since we got a request for it here we go:

```
528 \DeclareMathAccent{\mathring}{\mathalpha}{operators}{17}
```

#### 6.4.9 Radicals

```
529 \DeclareMathRadical{\sqrtsign}{symbols}{70}{largesymbols}{70}
```

#### 6.4.10 Over and under something, etc

```
530 \DeclareRobustCommand\overrightarrow[1]{\vbox{\m@th\ialign{##\crcr
531 \rightarrowfill\crcr\noalign{\kern-\p@\nointerlineskip}
532 $ \hfil\displaystyle{#1} \hfil$ \crcr}}
533 \DeclareRobustCommand\overleftarrow[1]{\vbox{\m@th\ialign{##\crcr
534 \leftarrowfill\crcr\noalign{\kern-\p@\nointerlineskip}
535 $ \hfil\displaystyle{#1} \hfil$ \crcr}}
536 \DeclareRobustCommand\overbrace[1]
537 {\mathop{\vbox{\m@th\ialign{##\crcr\noalign{\kern3\p@}%
538 \downbracefill\crcr\noalign{\kern3\p@\nointerlineskip}%
539 $ \hfil\displaystyle{#1} \hfil$ \crcr}}}\limits}
540 \DeclareRobustCommand\underbrace[1]{\mathop{\vtop{\m@th\ialign{##\crcr
541 $ \hfil\displaystyle{#1} \hfil$ \crcr
542 \noalign{\kern3\p@\nointerlineskip}%
543 \upbracefill\crcr\noalign{\kern3\p@}}}\limits}
```

(quite a waste of tokens, IMHO — Frank)

```
544 \DeclareRobustCommand\skew[3]
545 {{\muskip\z@#1mu\divide\muskip\z@\tw@ \mkern\muskip\z@
546 #2{\mkern-\muskip\z@{#3}\mkern\muskip\z@\mkern-\muskip\z@{}}}
547 \DeclareRobustCommand\rightarrowfill{$\m@th\smash-\mkern-7mu%
548 \cleaders\hbox{$\mkern-2mu\smash-\mkern-2mu$}\hfill
549 \mkern-7mu\mathord\rightarrow$}
550 \DeclareRobustCommand\leftarrowfill{$\m@th\mathord\leftarrow\mkern-7mu%
551 \cleaders\hbox{$\mkern-2mu\smash-\mkern-2mu$}\hfill
552 \mkern-7mu\smash-$}
553 \DeclareMathSymbol{\braceleft}{\mathord}{largesymbols}{7A}
554 \DeclareMathSymbol{\bracerd}{\mathord}{largesymbols}{7B}
555 \DeclareMathSymbol{\bracelu}{\mathord}{largesymbols}{7C}
556 \DeclareMathSymbol{\braceru}{\mathord}{largesymbols}{7D}
557 \DeclareRobustCommand\downbracefill{$\m@th \setbox\z@\hbox{$\braceleft$}%
558 \braceleft\leaders\vrule \height\ht\z@ \depth\z@\hfill\braceru
```

```

559 \braceleft\leaders\vrule \@height\ht\z@\@depth\z@\hfill\bracerd$}
560 \DeclareRobustCommand\upbracefill{$\mathop{\setbox\z@\hbox{$\braceleft$}}\limits^{\mathop{\setbox\z@\hbox{\bracerd}}\limits_{\mathop{\setbox\z@\hbox{\braceld}}\limits}}$}
561 \braceleft\leaders\vrule \@height\ht\z@\@depth\z@\hfill\bracerd
562 \braceld\leaders\vrule \@height\ht\z@\@depth\z@\hfill\braceru$}

```

#### 6.4.11 Delimiters

```

563 \DeclareMathDelimiter{\lmooustache} % top from (, bottom from)
564 {\mathopen}{largesymbols}{7A}{largesymbols}{40}
565 \DeclareMathDelimiter{\rmoustache} % top from), bottom from (
566 {\mathclose}{largesymbols}{7B}{largesymbols}{41}
567 \DeclareMathDelimiter{\arrowvert} % arrow without arrowheads
568 {\mathord}{symbols}{6A}{largesymbols}{3C}
569 \DeclareMathDelimiter{\Arrowvert} % double arrow without arrowheads
570 {\mathord}{symbols}{6B}{largesymbols}{3D}
571 \DeclareMathDelimiter{\Vert}
572 {\mathord}{symbols}{6B}{largesymbols}{0D}

\DeclareMathDelimiter produces a command that is robust (with an internal macro
containing the payload) so we should not use \let for making an alias
573 \%let\|=Vert
574 \DeclareMathDelimiter{\|}{%
575 {\mathord}{symbols}{6B}{largesymbols}{0D}}
576 \DeclareMathDelimiter{\vert}{%
577 {\mathord}{symbols}{6A}{largesymbols}{0C}}
578 \DeclareMathDelimiter{\uparrow}{%
579 {\mathrel}{symbols}{22}{largesymbols}{78}}
580 \DeclareMathDelimiter{\downarrow}{%
581 {\mathrel}{symbols}{23}{largesymbols}{79}}
582 \DeclareMathDelimiter{\updownarrow}{%
583 {\mathrel}{symbols}{6C}{largesymbols}{3F}}
584 \DeclareMathDelimiter{\Uparrow}{%
585 {\mathrel}{symbols}{2A}{largesymbols}{7E}}
586 \DeclareMathDelimiter{\Downarrow}{%
587 {\mathrel}{symbols}{2B}{largesymbols}{7F}}
588 \DeclareMathDelimiter{\Updownarrow}{%
589 {\mathrel}{symbols}{6D}{largesymbols}{77}}
590 \DeclareMathDelimiter{\backslash}{ % for double coset G\backslash H
591 {\mathord}{symbols}{6E}{largesymbols}{0F}}
592 \DeclareMathDelimiter{\rangle}{%
593 {\mathclose}{symbols}{69}{largesymbols}{0B}}
594 \DeclareMathDelimiter{\langle}{%
595 {\mathopen}{symbols}{68}{largesymbols}{0A}}
596 \DeclareMathDelimiter{\rbrace}{%
597 {\mathclose}{symbols}{67}{largesymbols}{09}}
598 \DeclareMathDelimiter{\lbrace}{%
599 {\mathopen}{symbols}{66}{largesymbols}{08}}
600 \DeclareMathDelimiter{\rceil}{%
601 {\mathclose}{symbols}{65}{largesymbols}{07}}
602 \DeclareMathDelimiter{\lceil}{%
603 {\mathopen}{symbols}{64}{largesymbols}{06}}
604 \DeclareMathDelimiter{\rfloor}{%
605 {\mathclose}{symbols}{63}{largesymbols}{05}}
606 \DeclareMathDelimiter{\lfloor}{%
607 {\mathopen}{symbols}{62}{largesymbols}{04}}

```

\lgroup There are three plain TeX delimiters which are not fully supported by NFSS, since they partly point into a bold cmr font. Allocating a full symbol font, just to have three delimiters seems a bit too much given the limited space available. For this reason only the extensible sizes are supported. If this is not desired one can use, without losing portability, define \mathbf and \mathtt as font symbol alphabet (setting up cmr/bx/n and cmtt/m/n as symbol fonts first) and modify the delimiter declarations to point with their small variant to those symbol fonts. (This is done in `oldlfont.dtx` so look there for examples.)

```

608 \DeclareMathDelimiter{\lgroup} % extensible (with sharper tips
609 {\mathopen}{largesymbols}{3A}{largesymbols}{3A}
610 \DeclareMathDelimiter{\rgroup} % extensible) with sharper tips
611 {\mathclose}{largesymbols}{3B}{largesymbols}{3B}
612 \DeclareMathDelimiter{\bracevert} % the vertical bar that extends braces
613 {\mathord}{largesymbols}{3E}{largesymbols}{3E}
```

(End definition for \lgroup, \rgroup, and \bracevert.)

## 6.5 Math versions of text commands

The \mathunderscore here is really a text definition, so it has been put back into `ltoutenc.dtx` (by Chris, 30/04/97) and should be removed from here.

These symbols are the math versions of text commands such as \P, \\$, etc.

\mathparagraph These math symbols are not in plain TeX.

```

\mathsection 614 \DeclareMathSymbol{\mathparagraph}{\mathord}{symbols}{7B}
\mathdollar 615 \DeclareMathSymbol{\mathsection}{\mathord}{symbols}{78}
\mathsterling 616 \DeclareMathSymbol{\mathdollar}{\mathord}{operators}{24}
\mathunderscore 617 \DeclareRobustCommand{\mathsterling}{\mathit{\mathchar"7024}}
618 \DeclareRobustCommand{\mathunderscore}{\kern.06em\vbox{\hrule\@width.3em}}
```

(End definition for \mathparagraph and others.)

\mathellipsis This is plain TeX's \ldots.

```
619 \DeclareRobustCommand{\mathellipsis}{\mathinner{\ldotp\ldotp\ldotp}}%
```

(End definition for \mathellipsis.)

## 6.6 Other special functions and parameters

### 6.6.1 Biggggg

```

620 </math>
621 <math | latexrelease>
622 <mathrelease>\IncludeInRelease{2018/12/01}%
623 <mathrelease> {\Big}{Start LR-mode}%
624 \DeclareRobustCommand{\big[1]{\leavevmode@ifvmode
625 {\hbox{$\left#1\vbox{to8.5p@{}\right.\n@space$}}}}
626 \DeclareRobustCommand{\Big[1]{\leavevmode@ifvmode
627 {\hbox{$\left#1\vbox{to11.5p@{}\right.\n@space$}}}}
628 \DeclareRobustCommand{\bigg[1]{\leavevmode@ifvmode
629 {\hbox{$\left#1\vbox{to14.5p@{}\right.\n@space$}}}}
630 \DeclareRobustCommand{\Bigg[1]{\leavevmode@ifvmode
631 {\hbox{$\left#1\vbox{to17.5p@{}\right.\n@space$}}}}
632 </math | latexrelease>
```

```

633 〈\latexrelease〉\EndIncludeInRelease
634 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
635 〈\latexrelease〉 {\Big}{Start LR-mode}%
636 〈\latexrelease〉\def\big#1{{\hbox{$\left#1\vbox{to8.5\p@{}\right.\n@space$}$}}}
637 〈\latexrelease〉\def\Big#1{{\hbox{$\left#1\vbox{to11.5\p@{}\right.\n@space$}$}}}
638 〈\latexrelease〉\def\bigg#1{{\hbox{$\left#1\vbox{to14.5\p@{}\right.\n@space$}$}}}
639 〈\latexrelease〉\def\Bigg#1{{\hbox{$\left#1\vbox{to17.5\p@{}\right.\n@space$}$}}}
640 〈\latexrelease〉\EndIncludeInRelease
641 〈*math〉
642 \def\n@space{\nulldelimiterspace\z@\m@th}

```

### 6.6.2 The log-like functions

\operator@font The \operator@font determines the symbol font used for log-like functions.

```

643 \def\operator@font{\mathgroup\symoperators}

```

(*End definition for \operator@font.*)

### 6.6.3 Parameters

```

644 \thinmuskip=3mu
645 \medmuskip=4mu plus 2mu minus 4mu
646 \thickmuskip=5mu plus 5mu

```

This finishes the low-level setup in `fontmath.ltx`.

```

647
```

## 7 Default cfg files

We provide default cfg files here to ensure that on installations that search large file trees we do not pick up some strange customisation files from somewhere.

```

648 <*cfgtext | cfgmath | cfgprel>
649 %%
650 %%
651 %%
652 %% Load the standard setup:
653 %%
654 <+cfgtext> \input{fonttext.ltx}
655 <+cfgmath> \input{fontmath.ltx}
656 <+cfgprel> \input{preload.ltx}
657 %%
658 %% Small changes could go here; see documentation in cfgguide.tex for
659 %% allowed modifications.
660 %%
661 %% In particular it is not allowed to misuse this configuration file
662 %% to modify internal LaTeX commands!
663 %%
664 %% If you use this file as the basis for configuration please change
665 %% the \ProvidesFile lines to clearly identify your modification, e.g.,
666 %%
667 <+cfgtext>%> \ProvidesFile{fonttext.cfg}[2001/06/01
668 <+cfgmath>%> \ProvidesFile{fonttext.cfg}[2001/06/01
669 <+cfgprel>%> \ProvidesFile{preload.cfg}[2001/06/01
670 %% Customised local font setup]
671 %%

```

672 %%  
673 ⟨/cfgtext | cfgmath | cfgprel⟩

# File C

## preload.dtx

### 1 Overview

This file contains a number of possible settings for preloading fonts during installation of NFSS2 (which is used by L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub> ). It will be used to generate the following files:

|                 |                                                                |
|-----------------|----------------------------------------------------------------|
| preload.min     | minimal subset of fonts necessary to run NFSS2                 |
| preload.ori     | preload of CM fonts similar to the old <code>1fonts.tex</code> |
| preload.ltx     | The standard selection of preloads                             |
| cmpreload.a.xpt | preload of CM fonts for 10pt document size                     |
| cmpreload.a.xip | preload of CM fonts for 11pt document size                     |
| cmpreload.a.xii | preload of CM fonts for 12pt document size                     |
| dcpreload.a.xpt | preload of DC fonts for 10pt size                              |
| dcpreload.a.xip | preload of DC fonts for 11pt size                              |
| dcpreload.a.xii | preload of DC fonts for 12pt size                              |

These files are for installations that make use of Computer Modern fonts either old encoding (OT1) or Cork encoding (T1). The Computer Modern fonts with Cork encoding are known as DC-fonts.

Most important is `preload.ltx` which is used during format generation. You are *not* allowed to change this file.

### 2 Customization

You can customize the preloaded fonts in your L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub>  system by installing a file with the name `preload.cfg`. If this file exists it will be used in place of the system file `preload.ltx`. You can, for example, copy one of the files mentioned above (that can be generated from this source) to `preload.cfg`.

Or you can define completely other preloads. In that case start from `preload.min` since that contains the fonts that have to be preloaded by \*all\* L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub>  systems.

Avoid using `preload.ori`, it will load so many fonts that on most installations it is nearly impossible to load other font families afterwards. This file is only generated to show what fonts have been preloaded by L<sup>A</sup>T<sub>E</sub>X 2.09.

If you normally use other fonts than Computer Modern `preload.min` might be best.

**Warning:** If you preload fonts with encodings other than the normally supported encodings you have to declare that encoding in a `fontdef.cfg` configuration file (see the documentation in the file `fontdef.dtx`). Adding an extra encoding to the format might produce non-portable documents, thus this should be avoided if possible.

### 3 Module switches for the DOCSTRIP program

The DOCSTRIP will generate the above file from this source using the following module directives:

|         |                                                         |
|---------|---------------------------------------------------------|
| driver  | produce a documentation driver file                     |
| preload | produce a preload... file                               |
| cm      | for OT1 encoded Computer Modern                         |
| dc      | for T1 encoded Computer Modern                          |
| min     | produce minimal subset                                  |
| xpt     | produce 10pt preloads                                   |
| xipt    | produce 11pt preloads                                   |
| xiipt   | produce 12pt preloads                                   |
| ori     | produce preloads similar to old <code>lfonts.tex</code> |
| tex     | produce preload.ltx                                     |

A typical DOCSTRIP command file would then have entries like:

```
generateFile{preload.min}{t}{\from{preload.dtx}{preload,min}}
```

for generating preload files.

## 4 A driver for this document

The next bit of code contains the documentation driver file for  $\text{\TeX}$ , i.e., the file that will produce the documentation you are currently reading. It will be extracted from this file by the DOCSTRIP program.

```
1 {*driver}
2 \documentclass{ltxdoc}
3 %\OnlyDescription % comment out for implementation details
4 \begin{document}
5 \DocInput{preload.dtx}
6 \end{document}
7
```

## 5 The code

We begin by loading the math extension font (`cmex10`) and the  $\text{\TeX}$  line and circle fonts. It is necessary to do this explicitly since these are used by the  $\text{\TeX}$  format. Since the internal font name contains / characters and digits we construct the name via `\csname`. These are the only fonts (!) that must be loaded in this file.

All `\DeclarePreloadSizes` can be removed or others can be added, they only influence the processing speed.

```
8 \expandafter\font\csname OMX/cmex/m/n/10\endcsname=cmex10\relax
9 \font\tenln =line10 \font\tenlnw =linew10\relax
10 \font\tencirc=lcircle10 \font\tencircw=lcirclew10\relax
```

The above fonts should not be touched but anything below this point here in the preload suggestions can be modified without any problems.

```
11 {-tex}*****%
12 {-tex}% Start any modification below this point **
13 {-tex}*****%
14 {-tex}%
15 %%%
16 %% Computer Modern Roman:
17 %%-----
```

```

18 <*ori>
19 \DeclarePreloadSizes{OT1}{cmr}{m}{n}
20 {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74,24.88}
21 \DeclarePreloadSizes{OT1}{cmr}{bx}{n}{9,10,10.95,12,14.4,17.28}
22 \DeclarePreloadSizes{OT1}{cmr}{m}{sl}{10,10.95,12}
23 \DeclarePreloadSizes{OT1}{cmr}{m}{it}{7,8,9,10,10.95,12}
24 </ori>
25 <+xpt & cm> \DeclarePreloadSizes{OT1}{cmr}{m}{n}{5,7,10}
26 <+xpt & dc> \DeclarePreloadSizes{T1}{cmr}{m}{n}{5,7,10}
27 <+xipt & cm> \DeclarePreloadSizes{OT1}{cmr}{m}{n}{6,8,10.95}
28 <+xipt & dc> \DeclarePreloadSizes{T1}{cmr}{m}{n}{6,8,10.95}
29 <+xiipt & cm> \DeclarePreloadSizes{OT1}{cmr}{m}{n}{6,8,12}
30 <+xiipt & dc> \DeclarePreloadSizes{T1}{cmr}{m}{n}{6,8,12}
31 %%
32 %% Computer Modern Sans:
33 %-----%
34 <+ori> \DeclarePreloadSizes{OT1}{cmss}{m}{n}{10,10.95,12}
35 %%
36 %% Computer Modern Typewriter:
37 %-----%
38 <+ori> \DeclarePreloadSizes{OT1}{cmtt}{m}{n}{9,10,10.95,12}
39 %%
40 %% Computer Modern Math:
41 %-----%
42 <*ori>
43 \DeclarePreloadSizes{OML}{cmm}{m}{it}
44 {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74}
45 \DeclarePreloadSizes{OMS}{cmsy}{m}{n}
46 {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74}
47 </ori>

```

The math fonts are the same for both DC and CM fonts. So far there isn't an agreed on standard.

```

48 <*xpt>
49 \DeclarePreloadSizes{OML}{cmm}{m}{it}{5,7,10}
50 \DeclarePreloadSizes{OMS}{cmsy}{m}{n}{5,7,10}
51 </xpt>
52 <*xipt>
53 \DeclarePreloadSizes{OML}{cmm}{m}{it}{6,8,10.95}
54 \DeclarePreloadSizes{OMS}{cmsy}{m}{n}{6,8,10.95}
55 </xipt>
56 <*xiipt>
57 \DeclarePreloadSizes{OML}{cmm}{m}{it}{6,8,12}
58 \DeclarePreloadSizes{OMS}{cmsy}{m}{n}{6,8,12}
59 </xiipt>
60 %%
61 %% LaTeX symbol fonts:
62 %-----%
63 <*ori>
64 \DeclarePreloadSizes{U}{lasy}{m}{n}
65 {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74}
66 </ori>
67 </preload>

```

# File D

## ltfntcmd.dtx

### Abstract

The commands defined in this file `ltfntcmd` are part of the kernel code for L<sup>A</sup>T<sub>E</sub>X 2<sub>&</sub>/NFSS2.

It is also meant to serve as documentation for package writers since it demonstrates how to define high-level font changing commands using a small number of creator functions.

### 1 Introduction

Font changes such as `\bfseries`, `\sffamily`, etc. are declarations; this means that their scope is delimited by the grouping structure, either by the next `\end` of some environment or by explicitly using a group, e.g., writing something like `{\bfseries...}` in the source. If you make the mistake of writing `\bfseries{...}` (thinking of `\bfseries` as a command with one argument) then the result is rather striking.

Font declarations are an artifact of the T<sub>E</sub>X system and for several reasons it is better to avoid them on the user level whenever possible. In L<sup>A</sup>T<sub>E</sub>X3 they will probably all be replaced by environments and by font commands taking one argument.

This file defines a creator function for such declarative font switches. This function creates commands which can be used in both math and text.

This file also defines a number of high-level commands (all starting with `\text...`) that have one argument and typeset this argument in the requested way. Thus these commands are for typesetting short pieces of text in a specific family, series or shape. These are all produced as examples of the use of a creator function which is itself also defined in this file.

Table 1 shows all these high-level commands in action. A further advantage of using these commands is that they automatically take care of any necessary italic correction on either side of their argument.

Thus, when using such commands, one does not have to worry about forgetting the italic correction when changing fonts. Only in very few situations is this additional space wrong but, for example, most typographers recommend omitting the italic correction if a small punctuation character, like a comma, directly follows the font change. Since the amount of correction required is partly a matter of taste, you can define in what situations the italic correction should be suppressed. This is done by putting the characters that should cancel a preceding italic correction in the list `\nocorrlist`.<sup>30</sup> The default definition for this list is produced by the following.

```
\newcommand \nocorrlist {,.}
```

It is best to declare the most often used characters first, because this will make the processing slightly faster. For example,

```
\emph{When using the \NFSS{} high-level commands,
the \emph{proper} use of italic corrections is
automatically taken care of}. Only
```

---

<sup>30</sup>Any package that changes the `\catcode` of a character inside `\nocorrlist` must then explicitly reset the list. Otherwise the changed character will no longer be recognized by the suppression algorithm.

| <i>Command</i>               | <i>Corresponds to</i>    | <i>Action</i>                                      |
|------------------------------|--------------------------|----------------------------------------------------|
| <code>\textnormal{..}</code> | <code>\normalfont</code> | Typeset argument in normal family                  |
| <code>\textrm{..}</code>     | <code>\rmfamily</code>   | Typeset argument in roman family                   |
| <code>\textsf{..}</code>     | <code>\sffamily</code>   | Typeset argument in <code>sans serif</code> family |
| <code>\texttt{..}</code>     | <code>\ttfamily</code>   | Typeset argument in <code>typewriter</code> family |
| <code>\textmd{..}</code>     | <code>\mdseries</code>   | Typeset argument in medium series                  |
| <code>\textbf{..}</code>     | <code>\bfseries</code>   | Typeset argument in <b>bold</b> series             |
| <code>\textup{..}</code>     | <code>\upshape</code>    | Typeset argument in normal shape                   |
| <code>\textit{..}</code>     | <code>\itshape</code>    | Typeset argument in <i>italic</i> shape            |
| <code>\textsl{..}</code>     | <code>\slshape</code>    | Typeset argument in <i>slanted</i> shape           |
| <code>\textsc{..}</code>     | <code>\scshape</code>    | Typeset argument in <code>SMALL CAPS</code> shape  |
| <code>\emph{..}</code>       | <code>\em</code>         | Typeset argument <i>emphasized</i>                 |

Table 1: Font-change commands with arguments

The font change commands provided here all start with `\text..` to emphasize that they are for use in normal text and to be easily memorable. They automatically take care of any necessary italic correction on either side of the argument.

`\emph{sometimes}` one has to help `\LaTeX{}` by adding a `\verb=\nocorr=` command.

which results in:

*When using the NFSS high-level commands, the proper use of italic corrections is automatically taken care of. Only sometimes one has to help L<sup>A</sup>T<sub>E</sub>X by adding a `\nocorr` command.*

In contrast, the use of the declaration forms is often more appropriate when you define your own commands or environments.

```
\newenvironment{bfitemize}{\begin{itemize}\normalfont\bfseries}
{\end{itemize}}
\begin{bfitemize}
\item This environment produces boldface items.
\item It is defined in terms of \LaTeX's
 \texttt{itemize} environment and NFSS
 declarations.
\end{bfitemize}
```

This gives:

- This environment produces boldface items.
- It is defined in terms of L<sup>A</sup>T<sub>E</sub>X's `itemize` environment and NFSS declarations.

In addition to global customization of when to insert the italic correction, it is of course sometimes necessary to explicitly insert one with `\!/`.

It is also possible to suppress the italic correction in individual instances. For this, the command `\nocorr` is provided.

The `\nocorr` must appear as the first or last token inside the braces of the argument of the `\text...` commands, at that end of the text where you wish to suppress the italic correction.

It is worth pointing out here that inserting a `\V` in places where it can have no function (i.e. anywhere except immediately after a slanted letter) is not an error—it will just be silently ignored. Unfortunately this is not true if the redefinition of `\V` in `amstex.sty` is used as this version can cause space to be removed immediately before the `\V`.

## 2 The implementation

`\DeclareTextFontCommand` This is the creator function for `\text..` commands. It gives a warning if `\foo` or `\fragfoo` is already defined.

In math mode it simply puts the font declaration and text into a box (possibly an automagically sized one).

Otherwise it first scans the text to see where `\nocorr` occurs within it. This sets the `\check@ic` commands to do what is necessary concerning the italic correction at both ends.

The algorithm for deciding whether to put in an italic correction is not very subtle: one is added whenever the newly current font is not itself positively sloped, unless the next token is a character in the ‘nocorr’ list. At the end of the text this is done after closing the group so as to check the ‘outer font’. Note that this will often result in adding an italic correction token after a character in an unsloped font; we believe (in early 2003) that this is perhaps inefficient but not dangerous.

It also now checks for empty contents of the text command and optimizes this case. Some care is also taken to check that doing dangerous things in vertical mode is avoided.

The italic correction token is added to the horizontal list before (in the list) an immediately preceding non-zero glob of glue (skip) and any non-zero penalty preceding that since, in the typical case, this puts it immediately after the last character in the preceding word.

Note that it is necessary to put in the `\aftergroup\maybe@ic` at the end of the group so that it comes after any other aftergroup tokens and immediately before the following tokens. It is also necessary to remove the `\fi` from the token list before the group ends; this is done by adding an `\expandafter` just before the closing brace.

```
1 {*2ekernel}
2 \def \DeclareTextFontCommand #1#2{%
3 \DeclareRobustCommand#1[1]{%
4 \ifmmode
5 \nfss@text{#2##1}%
6 \else
7 \hmode@bgroup
8 \text@command{##1}%
9 #2\check@icl ##1\check@icr
10 \expandafter
11 \egroup
12 \fi
13 }%
14 }
```

(End definition for `\DeclareTextFontCommand`.)

`\textrm` Now we define the `\text<family>` commands in terms of the above; `\textttt` does not look very nice!

`\textsf` 15 `\DeclareTextFontCommand{\textrm}{\rmfamily}`  
`\textnormal` 16 `\DeclareTextFontCommand{\textsf}{\sffamily}`  
17 `\DeclareTextFontCommand{\textttt}{\ttfamily}`  
18 `\DeclareTextFontCommand{\textnormal}{\normalfont}`

(End definition for `\textrm` and others.)

`\textbf` For the series attribute:

`\textmd` 19 `\DeclareTextFontCommand{\textbf}{\bfseries}`  
20 `\DeclareTextFontCommand{\textmd}{\mdseries}`

(End definition for `\textbf` and `\textmd`.)

`\textit` And for the shapes:

`\textsl` 21 `\DeclareTextFontCommand{\textit}{\itshape}`  
`\textsc` 22 `\DeclareTextFontCommand{\textsl}{\slshape}`  
`\textup` 23 `\DeclareTextFontCommand{\textsc}{\scshape}`  
24 `\DeclareTextFontCommand{\textup}{\upshape}`

(End definition for `\textit` and others.)

`textulc`  
`textsw` 25 `/2ekernel`  
`textssc` 26 `(*2ekernel | latexrelease)`  
27 `\langle latexrelease \rangle \IncludeInRelease{2020/02/02} %`  
28 `\langle latexrelease \rangle \textulc{Additional text commands} %`  
29 `\DeclareTextFontCommand{\textulc}{\ulcshape}`  
30 `\DeclareTextFontCommand{\textsw}{\swshape}`  
31 `\DeclareTextFontCommand{\textssc}{\sscshape}`  
32 `\langle /2ekernel | latexrelease \rangle`  
33 `\langle latexrelease \rangle \EndIncludeInRelease`  
34 `\langle latexrelease \rangle \IncludeInRelease{0000/00/00} %`  
35 `\langle latexrelease \rangle \textulc{Additional text commands} %`  
36 `\langle latexrelease \rangle`  
37 `\langle latexrelease \rangle \let\textulc\@undefined`  
38 `\langle latexrelease \rangle \let\textsw\@undefined`  
39 `\langle latexrelease \rangle \let\textssc\@undefined`  
40 `\langle latexrelease \rangle \EndIncludeInRelease`  
41 `\langle *2ekernel \rangle`

(End definition for `textulc`, `textsw`, and `textssc`.)

`\emph` Finally we have the `\em` font change declaration of L<sup>A</sup>T<sub>E</sub>X. The corresponding definition with argument is

42 `\DeclareTextFontCommand{\emph}{\em}`

(End definition for `\emph`.)

`\nocorr` This is just a label, so it does nothing; it should also be unexpandable.

43 `\let\nocorr\relax`

(End definition for \nocorr.)

\check@icl We define these defaults in case some error causes them to be expanded at the wrong time.

```
44 \let \check@icl \@empty
45 \let \check@icr \@empty
```

(End definition for \check@icl and \check@icr.)

\text@command This checks for a \nocorr as the first token in its argument and also for one in any other \check@nocorr@ position not protected within braces (the latter is treated as if it were at the end of the argument).

Is this the correct action in the ‘empty’ case? It is efficient but typographically it is, strictly, incorrect!

```
46 \def \text@command #1{%
47 \edef \reserved@a {\unexpanded{#1}}%
48 \ifx \reserved@a \@empty
49 \let \check@icl \@empty
50 \let \check@icr \@empty
51 \else
```

\space is a reserved word in L<sup>A</sup>T<sub>E</sub>X or actually already in plain T<sub>E</sub>X. If somebody really redefines it so many things will break that I don’t see any reason to make this routine here slower than necessary.

```
52 % \def \reserved@b { }%
53 % \ifx \reserved@a \reserved@b
54 \ifx \reserved@a \space
55 \let \check@icl \@empty
56 \let \check@icr \@empty
57 \else
58 \check@nocorr@ #1\nocorr@nil
59 \fi
60 \fi
61 }
62 \def \check@nocorr@ #1#2\nocorr#3@nil {%
```

The two checks are initialised here to their values in the normal case.

```
63 \let \check@icl \maybe@ic
64 \def \check@icr {\ifvmode \else \aftergroup \maybe@ic \fi}%
65 \def \reserved@a {\nocorr}%
66 \def \reserved@b {#1}%
67 \def \reserved@c {#3}%
68 \ifx \reserved@a \reserved@b
69 \ifx \reserved@c \@empty
```

In this case there is a \nocorr at the start but not at the end, so \check@icl should be empty.

```
70 \let \check@icl \@empty
71 \else
```

Otherwise there is a \nocorr both at the start and elsewhere, so no italic corrections should be added.

```
72 \let \check@icl \@empty
73 \let \check@icr \@empty
74 \fi
```

```

75 \else
76 \ifx \reserved@c \empty
```

In this case there is no `\nocorr` anywhere, so we need to check for an italic correction at both the beginning and the end. This has been set up as the default so no code is needed here.

```

77 \else
```

In this case there is no `\nocorr` at the start but there is one elsewhere, so no `\aftergroup` is needed.

```

78 \let \check@icr \empty
79 \fi
80 \fi
81 }
```

*(End definition for `\text@command` and `\check@nocorr@`.)*

`\ifmaybe@ic` Switch used solely within `\maybe@ic` not interfering with other switches.

```

82 \newif\ifmaybe@ic
```

*(End definition for `\ifmaybe@ic`.)*

`\maybe@ic` These macros implement the italic correction.

```

\maybe@ic@ 83 \def \maybe@ic {\futurelet\@let@token\maybe@ic@}
84 \def \maybe@ic@ {\%
```

We first check to see if the current font is positively sloped. (But do not forget the message Rainer sent about an upright font with non-zero slope! Or is this an urban myth?) It has been suggested that this should test against a small positive value, but what?

```

85 \ifdim \fontdimen@ne\font>\z@
86 \else
87 \maybe@icttrue
```

It would be possible, but probably not worthwhile, to continue the forward scan beyond any closing braces.

```

88 \expandafter\@tfor\expandafter\reserved@a\expandafter:\expandafter=%
89 \nocorrlist
```

We have to hide the `\@let@token` in the macro `\t@st@ic` rather than testing it directly in the loop since it might be `\let` to a `\fi` or `\else`, which would result in chaos.

```

90 \do \t@st@ic
```

Frank thinks that the next bit is inefficient if done after the second change. Chris thinks that most all of this is inefficient for the commonest cases: but that is the price of a cleverer algorithm. It is certainly needed to deal with the use of `\nolinebreak`.

```

91 \ifmaybe@ic \sw@slant \fi
92 \fi
93 }
```

*(End definition for `\maybe@ic` and `\maybe@ic@`.)*

`\t@st@ic` The next token in the input stream is stored in `\@let@token` via a `\let`, the current token from `\nocorrlist` is stored via `\def` in `\reserved@a`. To compare them we have to fiddle around a bit.

If the only things to check were characters then this could be done via an `\if` thus their catcodes would not matter; but this will not work whilst `\futurelet` is used above.

```

94 \def \t@st@ic {%
95 \expandafter\let\expandafter\reserved@b\expandafter=\reserved@a\relax
96 \ifx\reserved@b\@let@token

```

If they are the same we record the fact and jump out of the loop.

```

97 \maybe@icfalse
98 \break@tfor
99 \fi
100 }

```

(End definition for `\t@st@ic`.)

`\sw@slant` The definition of the mysterious `\sw@slant` command is as follows.

```

101 \def \sw@slant {%

```

It is surely correct to put in an italic correction when there is no skip. If the last thing on the list is actually a zero skip (including things whose dimension part is zero, such as `\hfill`), or anything other than a character, then the italic correction will have no effect.

In order to work correctly with unbreakable spaces from `\~` (and other common forms of line-breaking control) we also move back across a penalty before the glue.

```

102 \ifdim \lastskip=z@
103 \fix@penalty
104 \else
105 \skip@\lastskip
106 \unskip
107 \fix@penalty
108 \hskip \skip@
109 \fi
110 }

```

The above code means: “If there is a non-zero space just before the current position (`\ifdim...`) save the amount of that space (`\skip@\lastskip`), remove it (`\unskip`), then do a similar thing if there is a penalty just before the skip, and finally put the space back in.”

Since zero glue cannot be distinguished in this context from no glue, we dare not put in an `\hskip` in this case as this may produce an unwanted breakpoint. This is not satisfactory.

The penalty before the glue is handled similarly, with the same caveats concerning the zero case. Is this the first recorded use of `\unpenalty` in standard L<sup>A</sup>T<sub>E</sub>X code?

```

111 \def \fix@penalty {%
112 \ifnum \lastpenalty=z@
113 \@@italiccorr
114 \else
115 \count@\lastpenalty
116 \unpenalty
117 \@@italiccorr

```

```

118 \penalty \count0
119 \fi
120 }

```

(End definition for `\sw@slant` and `\fix@penalty`.)

- `\nocorrlist` This holds the list of characters that should prevent italic correction. They should be ordered by decreasing frequency of use. If any such character is made active later on one needs to redefine the list so that the active character becomes part of it.

```

121 \def \nocorrlist {,.}
122 (End definition for \nocorrlist.)

```

- `\nfss@text` This command will by default behave like a L<sup>A</sup>T<sub>E</sub>X `\mbox` but may be redefined by packages such as `amstext.sty` to be a bit cleverer.

```

122 \ifx \nfss@text \undefined
123 \def \nfss@text {\leavevmode\hbox}
124 \fi

```

(End definition for `\nfss@text`.)

- `\DeclareOldFontCommand` This is the function used to create declarative font-changing commands that can also be used to change alphabets in math-mode.

Usage: `\DeclareOldFontCommand \fn{\font-change decls} {math-alphabet}`

Here `\fn` is the font-declaration command being defined, `\font-change decls` is the declaration it will expand to in text-mode, and `{math-alphabet}` is the (single) math alphabet specifier which is to be used in math-mode.

It does not care whether the command being defined already exists but it does give a warning if it redefines anything.

Here are some typical examples of its use in conjunction with more basic NFSS2 font commands.

```

\DeclareOldFontCommand{\rm}{\normalfont\rmfamily}{\mathrm}
\DeclareOldFontCommand{\sf}{\normalfont\sfamily}{\mathrm{sf}}
\DeclareOldFontCommand{\tt}{\normalfont\ttfamily}{\mathrm{tt}}

```

```

125 \def \DeclareOldFontCommand #1#2#3{%
126 \ DeclareRobustCommand #1{\@fontswitch {#2}{#3}}%
127 }

```

(End definition for `\DeclareOldFontCommand`.)

- `\@fontswitch` These two commands actually do the necessary tests and declarative font- or alphabet-changing.

```

128 \def \@fontswitch #1#2{%
129 \ifmmode
130 \let \math@bgroup \relax
131 \def \math@egroup {\let \math@bgroup \math@bgroup
132 \let \math@egroup \math@egroup}%

```

We need to have a `\relax` in the following line in case the #2 is something like `\mathsf` grabbing the next token as an argument. For this reason the code also uses explicit arguments again (see pr/1275).

```
133 #2\relax
134 \else
135 #1%
136 \fi
137 }
138 \let \@@math@bgroup \math@bgroup
139 \let \@@math@egroup \math@egroup

(End definition for \fontswitch, \math@bgroup, and \math@egroup.)
```

These commands are available only in the preamble.

```
140 \onlypreamble \DeclareTextFontCommand
141 \onlypreamble \DeclareOldFontCommand
```

### 3 Initialization

`\normalsize` This is defined to produce an error.

```
142 \def\normalsize{%
143 \@latex@error {The font size command \protect\normalsize\space
144 is not defined:\MessageBreak
145 there is probably something wrong with
146 the class file}\@eha
147 }
148 ⟨/2ekernel⟩
```

(End definition for `\normalsize`.)

## File E

# lttextcomp.dtx

This file contains the implementation for accessing the glyphs provided by the TS1 encoding (Text Companion Encoding). This is now offered as part of the kernel and so the `textcomp` package which used to provide the definitions is now mainly needed for compatibility reasons (and doesn't do much any more).

```
1 {*2ekernel | latexrelease}
2 <latexrelease>\NewModuleRelease{2020/02/02}{lttextcomp}
3 <latexrelease> {Text Companion symbols}
```

\oldstylenums  
\legacyoldstylenums Preserve the old definition of `\oldstylenums` under a different name.

This macro implements old style numerals but only works if we assume that the standard math fonts are used. Thus it needs changing in case other math encodings are used.

```
4 \DeclareRobustCommand\legacyoldstylenums[1]{%
5 \begingroup
```

Provide spacing using the interword space of the current font.

```
6 \spaceskip\fontdimen\tw@\font
```

Then switch to the math italic font. We don't change the current value of `\f@series` which means that you can use bold numerals if `\bfseries` is in force. As family we use `\rmdefault` which means that this only works if there exist an OML encoded version of that font or rather a corresponding .fd file (which is the case for standard L<sup>A</sup>T<sub>E</sub>X fonts even though they only contain substitutions).

```
7 \usefont{OML}{\rmdefault}{\f@series}{it}%
8 \mathgroup\symletters #1%
9 \endgroup
10 }
```

And here is the improved one that adjusts depending on surroundings.

```
11 \DeclareRobustCommand\oldstylenums[1]{%
12 \begingroup
13 \ifmmode
14 \mathgroup\symletters #1%
15 \else
```

The `\CheckEncodingSubset` is discussed below.

```
16 \CheckEncodingSubset\use@text@encoding{TS1}\tc@oldstylesubst2{{#1}}%
17 \fi
18 \endgroup
19 }
```

The helper to select the substitution if needed.

```
20 \def\tc@oldstylesubst#1{%
21 \tc@errorwarn
22 {Oldstyle digits unavailable for
23 family \f@family.\MessageBreak
24 Default oldstyle digits used instead}\@eha
25 \bgroup
26 \expand@font@defaults
```

The substitution defaults are provided in the file `fonttext.ltx`.

```
27 \ifx\f@family\rmdef@ult
28 \fontfamily\rmsubstdefault
29 \else\ifx\f@family\sffamily\sfdef@ult
30 \fontfamily\sfsbstdefault
31 \else\ifx\f@family\ttdef@ult
32 \fontfamily\ttsbstdefault
33 \else
34 \fontfamily\textcompsubstdefault
35 \fi\fi\fi
36 \fontencoding{TS1}\selectfont#1%
37 \egroup
38 }
```

(End definition for `\oldstylenums` and `\legacyoldstylenums`.)

`\textcompsubstdefault` Here is the default for the “unknown” case:

```
39 \def\textcompsubstdefault{\rmsubstdefault}
```

(End definition for `\textcompsubstdefault`.)

`\DeclareEncodingSubset` The declaration takes 3 mandatory arguments: an *encoding* for which a subsetting is wanted (currently always TS1, and most likely forever), the *font family* for which we declare the subset and finally the *subset* number (between 0 (all of the encoding is supported) and 9 many glyphs are missing).

For TS1 the numbers have been chosen in a way that most fonts can be fairly correctly categorized, but the default settings are always conservative, that is they may claim that less glyphs are supported than there actually are.

As these days many font families are set up to end in -LF (lining figures), -OsF (oldstyle figures), etc. the declaration supports a shortcut: if the *font family* name ends in -\* then the star gets replaced by these common ending, e.g.,

```
\DeclareEncodingSubset{TS1}{Alegreya-*}{2}
```

is the same as writing

```
\DeclareEncodingSubset{TS1}{Alegreya-LF}{2}
\DeclareEncodingSubset{TS1}{Alegreya-OsF}{2}
\DeclareEncodingSubset{TS1}{Alegreya-TLF}{2}
\DeclareEncodingSubset{TS1}{Alegreya-T0sF}{2}
```

If only some are needed then one can define them individually but in many cases all four are wanted, hence the shortcut.

The coding of the declaration has no error checking as it is mostly for internal use.

```
40 \def\DeclareEncodingSubset#1#2{%
41 \DeclareEncodingSubset@aux{#1}#2*\DeclareEncodingSubset@aux
42 }
43 \def\DeclareEncodingSubset@aux#1#2*#3\DeclareEncodingSubset@aux#4{%
```

if #3 is empty then there was no star, otherwise we define all four variants.

```
44 \expandafter\ifx\expandafter X\detokenize{#3}X%
45 \@DeclareEncodingSubset{#1}{#2}{#4}%
46 \else
47 \@DeclareEncodingSubset{#1}{#2LF}{#4}%
48 \@DeclareEncodingSubset{#1}{#2TLF}{#4}%
49 \@DeclareEncodingSubset{#1}{#20sF}{#4}%
50 \@DeclareEncodingSubset{#1}{#2T0sF}{#4}%
51 \fi
52 }
```

The subset info is stored in a command with the name `\family:subset` so if that already exists we change otherwise declare a subset.

```
53 \def\@DeclareEncodingSubset#1#2#3{%
54 \@ifundefined{#1:#2}{%
55 {\@font@info{Setting #2 sub-encoding to #1/#3}}%
56 {\@font@info{Changing #2 sub-encoding to #1/#3}}%
57 \@namedef{#1:#2}{#3}}
```

Any reason to allow those in the middle of documents?

```
58 \@onlypreamble\DeclareEncodingSubset
59 \@onlypreamble\DeclareEncodingSubset@aux
60 \@onlypreamble\@DeclareEncodingSubset
```

(End definition for `\DeclareEncodingSubset`.)

`\CheckEncodingSubset` The command `\CheckEncodingSubset` will check if the current font family has the right encoding subset to typeset a certain command. It takes five arguments as follows: first argument is either `\UseTextSymbol`, `\UseTextAccent` depending on whether or not the symbol is a text symbol or a text accent.

The second argument is the encoding from which this symbol should be fetched.

The third argument is either a fake accessor command or an error message. the code in that argument (if ever executed) receives two arguments: #2 and #5 of `\CheckEncodingSubset`.

Argument four is the subset encoding id to test against: if this value is higher than the subset id of the current font family then we typeset the symbol, i.e., execute `#1{#2}#5` otherwise it runs #3#5, e.g., to produce an error message or fake the glyph somehow.

Argument five is the symbol or accent command that is being checked.

For usage examples see definitions below.

```
61 \def\CheckEncodingSubset#1#2#3#4#5{%
62 \ifnum #4>%
63 \expandafter\ifx\csname #2:\f@family\endcsname\relax
64 \csname #2:?\endcsname
65 \else
66 \csname #2:\f@family\endcsname
67 \fi
68 \relax
69 \expandafter\@firstoftwo
70 \else
71 \expandafter\@secondoftwo
72 \fi
73 {#1{#2}}{#3}%
74 #5%
75 }
```

(End definition for \CheckEncodingSubset.)

To set up the glyphs for the subsets we need a number helpers.

- \tc@errorwarn To we produce errors, warnings, or only info in the transcripts if glyphs require substitutions? By default it is “info” only. With the `textcomp` package that can be changed.

```
76 \def\tc@errorwarn#1#2{\@latex@info{#1}}
```

(End definition for \tc@errorwarn.)

\tc@subst

```
77 \def\tc@subst#1{%
78 \tc@errorwarn
79 {Symbol \string#1 not provided by\MessageBreak
80 font family \f@family\space
81 in TS1 encoding.\MessageBreak Default family used instead}\@eha
82 \bgroup
83 \expand@font@defaults
84 \ifx\f@family\rmdef@ult
85 \fontfamily\rmsubstdefault
86 \else\ifx\f@family\sfdef@ult
87 \fontfamily\sfsubstdefault
88 \else\ifx\f@family\ttdef@ult
89 \fontfamily\ttsubstdefault
90 \else
91 \fontfamily{textcompsubstdefault}
92 \fi\fi\fi}
```

Whatever default was chosen, we claim now (locally hopefully) that it can handle all slots (even if not true) to avoid looping in certain situations, e.g., when something was set up incorrectly.

```
93 \namedef{TS1:\f@family}{0}%
94 \selectfont#1%
95 \egroup
96 }
```

(End definition for \tc@subst.)

- \tc@fake@euro \tc@fake@euro is an example of a “fake” definition to use in arg #3 of the command `\CheckEncodingSubset` when a symbol is not available in a certain font family. Here we produce a poor man’s Euro symbol by combining a “C” with a “=“.

```
97 \def\tc@fake@euro#1{%
98 \leavevmode
99 \font@info{Faking \noexpand#1 for font family
100 \f@family\MessageBreak in TS1 encoding}%
101 \valign{##\cr
102 \vfil\hbox to 0.07em{\dimen@\f@size\p@
103 \math@fontsfalse
104 \fontsize{.7\dimen@}\z@\selectfont=\hss}%
105 \vfil\cr%
106 \hbox{C}\crcr
107 }%
108 }
```

(End definition for \tc@fake@euro.)

\tc@check@symbol These are two abbreviations that we use below to check symbols and accents in TS1.  
\tc@check@accent Only there to save some space, e.g., we can then write

```
DeclareTextCommandDefault{\textcurrency}{\tc@check@symbol3\textcurrency}
```

to ensure that \textcurrency is only typeset if the current font has a TS1 subset id of less than 3. Otherwise \tc@error is called telling the user that for this font family \textcurrency is not available.

```
109 \def\tc@check@symbol{\CheckEncodingSubset\UseTextSymbol{TS1}\tc@subst}
```

Accents have been made an error in the textcomp package when not available. Now that we provide the functionality in the kernel we avoid the error by swapping in a T1 accent if the TS1 accent is not available.

```
110 \%def\tc@check@accent{\CheckEncodingSubset\UseTextAccent{TS1}\tc@error}
111 \def\tc@check@accent#1{\CheckEncodingSubset\UseTextAccent
112 \tc@swap@accent#1}
113 \def\tc@swap@accent#1#2{\UseTextAccent{T1}#1}
```

(End definition for \tc@check@symbol and \tc@check@accent.)

## 1 Sub-encodings

Here are the default definitions for the TS1 symbols. First those that we assume are always available if a font implements TS1.

```
114 \DeclareTextSymbolDefault{\textdollar}{TS1}
115 \UndeclareTextCommand{\textdollar}{OT1} % don't use the OT1 def any longer
116 \DeclareTextSymbolDefault{\textsterling}{TS1}
117 \UndeclareTextCommand{\textsterling}{OT1} % don't use the OT1 def any longer
118 \DeclareTextSymbolDefault{\textperthousand}{TS1}
119 \UndeclareTextCommand{\textperthousand}{T1} % don't use the T1 def
```

Using \UndeclareTextCommand above is enough only if the encoding definition files are not reloaded afterwards. In the past that happened if `fontenc` was used in the document preamble (not any longer). So in some sense it is better to fully remove them from the encoding files, but for rollbacks it is easier to keep them in for now.

These are the standard itemize and footnote symbols originally taken from OMS and now from TS1:

```
120 \DeclareTextSymbolDefault{\textasteriskcentered}{TS1}
121 \DeclareTextSymbolDefault{\textbullet}{TS1}
122 \DeclareTextSymbolDefault{\textdaggerdbl}{TS1}
123 \DeclareTextSymbolDefault{\textdagger}{TS1}
124 \DeclareTextSymbolDefault{\textparagraph}{TS1}
125 \DeclareTextSymbolDefault{\textperiodcentered}{TS1}
126 \DeclareTextSymbolDefault{\textsection}{TS1}
```

And here are the other TS1 glyphs that are implemented by every font (or nearly every—a few are commented out and moved to sub-encoding 9, because they aren't around in some fonts).

```
127 %%\DeclareTextSymbolDefault{\textbardbl}{TS1} % subst in sub-enc 9 above
128 \DeclareTextSymbolDefault{\textbrokenbar}{TS1}
129 %%\DeclareTextSymbolDefault{\textcelsius}{TS1} % subst in sub-enc 9 above
130 \DeclareTextSymbolDefault{\textcent}{TS1}
```

```

131 \DeclareTextSymbolDefault{\textcopyright}{TS1}
132 \DeclareTextSymbolDefault{\textdegree}{TS1}
133 \DeclareTextSymbolDefault{\textdiv}{TS1}
134 \DeclareTextSymbolDefault{\textlnot}{TS1}
135 \DeclareTextSymbolDefault{\textonehalf}{TS1}
136 \DeclareTextSymbolDefault{\textonequarter}{TS1}
137 %%\DeclareTextSymbolDefault{\textonesuperior}{TS1} % subst in sub-enc 9 above
138 \DeclareTextSymbolDefault{\textordfeminine}{TS1}
139 \DeclareTextSymbolDefault{\textordmasculine}{TS1}
140 \DeclareTextSymbolDefault{\textpm}{TS1}
141 \DeclareTextSymbolDefault{\textquotesignle}{TS1}
142 \DeclareTextSymbolDefault{\textquotestraightbase}{TS1}
143 \DeclareTextSymbolDefault{\textquotestraightdblbase}{TS1}
144 \DeclareTextSymbolDefault{\textregistered}{TS1}
145 %%\DeclareTextSymbolDefault{\textthreequartersemdash}{TS1} % subst in sub-enc 9 above
146 \DeclareTextSymbolDefault{\textthreequarters}{TS1}
147 %%\DeclareTextSymbolDefault{\textthreesuperior}{TS1} % subst in sub-enc 9 above
148 \DeclareTextSymbolDefault{\texttimes}{TS1}
149 \DeclareTextSymbolDefault{\texttrademark}{TS1}
150 %%\DeclareTextSymbolDefault{\texttwelveudash}{TS1} % subst in sub-enc 9 above
151 %%\DeclareTextSymbolDefault{\texttwosuperior}{TS1} % subst in sub-enc 9 above
152 \DeclareTextSymbolDefault{\textyen}{TS1}
153 \DeclareTextSymbolDefault{\textcapitalcompwordmark}{TS1}
154 \DeclareTextSymbolDefault{\textascendercompwordmark}{TS1}

```

In the following sections the remaining default definitions are ordered by sub-encoding in which they are become **unavailable**, i.e., they are not provided in the sub-encoding with that number and all sub-encodings with higher numbers.

Thus the symbols that are available in sub-encoding  $x$  are the symbols above (always available) and the symbols listed as becoming unavailable in sub-encodings  $x + 1$  and higher.

## 1.1 Unavailable in sub-encoding 1 and higher (drop symbols not working in Latin Modern)

The `\textcircled` is available but the glyph is simply too small so we keep using the OMS glyph.

```

155 \DeclareTextCommandDefault{\textcircled}
156 {\CheckEncodingSubset\UseTextAccent{TS1}{\UseTextAccent{OMS}}1\textcircled}

```

## 1.2 Unavailable in sub-encoding 2 (majority of new OTF fonts via autoinst) and higher

```

157 \DeclareTextCommandDefault{\t}
158 {\CheckEncodingSubset\UseTextAccent{TS1}{\UseTextAccent{OML}}2\t}

```

Capital accents are really only very seldom implemented, so from sub-encoding 2 onwards we use the normal T1 accents if they are asked for in the document.

In Unicode engines we don't implement them at all but always use the basic accents instead. whether that works or not really depends on the font, something like `\"X` usually comes out wrong in Unicode engines.

```

159 \ifx\Umathcode\@undefined
160 \DeclareTextCommandDefault{\capitalacute}

```

```

161 {\tc@check@accent{'}}2\capitalacute}
162 \DeclareTextCommandDefault{\capitalbreve}
163 {\tc@check@accent{\u}2\capitalbreve}
164 \DeclareTextCommandDefault{\capitalcaron}
165 {\tc@check@accent{\v}2\capitalcaron}
166 \DeclareTextCommandDefault{\capitalcedilla}
167 {\tc@check@accent{\c}2\capitalcedilla}
168 \DeclareTextCommandDefault{\capitalcircumflex}
169 {\tc@check@accent{\^}2\capitalcircumflex}
170 \DeclareTextCommandDefault{\capitaldieresis}
171 {\tc@check@accent{\"}2\capitaldieresis}
172 \DeclareTextCommandDefault{\capitaldotaccent}
173 {\tc@check@accent{\.}2\capitaldotaccent}
174 \DeclareTextCommandDefault{\capitalgrave}
175 {\tc@check@accent{\`}2\capitalgrave}
176 \DeclareTextCommandDefault{\capitalhungarumlaut}
177 {\tc@check@accent{\H}2\capitalhungarumlaut}
178 \DeclareTextCommandDefault{\capitalmacron}
179 {\tc@check@accent{\=}{2\capitalmacron}}
180 \DeclareTextCommandDefault{\capitalogonek}
181 {\tc@check@accent{\k}2\capitalogonek}
182 \DeclareTextCommandDefault{\capitalring}
183 {\tc@check@accent{\r}2\capitalring}
184 \DeclareTextCommandDefault{\capitaltie}
185 {\tc@check@accent{\t}2\capitaltie}
186 \DeclareTextCommandDefault{\capitaltilde}
187 {\tc@check@accent{\~}2\capitaltilde}

```

For \newtie and \capitalnewtie this is actually wrong, they should pick up the accent from the substitution font (not done yet).

```

188 \DeclareTextCommandDefault{\newtie}
189 {\tc@check@accent{\t}2\newtie}
190 \DeclareTextCommandDefault{\capitalnewtie}
191 {\tc@check@accent{\t}2\capitalnewtie}

```

In Unicode engines we just execute the simple accents:

```

192 \else
193 \DeclareTextCommandDefault{\capitalacute{@tabacckludge'}}
```

~~\DeclareTextCommandDefault{\capitalbreve{\u}}~~
~~\DeclareTextCommandDefault{\capitalcaron{\v}}~~
~~\DeclareTextCommandDefault{\capitalcedilla{\c}}~~
~~\DeclareTextCommandDefault{\capitalcircumflex{\^}}~~
~~\DeclareTextCommandDefault{\capitaldieresis{\\"}}~~
~~\DeclareTextCommandDefault{\capitaldotaccent{\.}}~~
~~\DeclareTextCommandDefault{\capitalgrave{@tabacckludge'}}~~
~~\DeclareTextCommandDefault{\capitalhungarumlaut{\H}}~~
~~\DeclareTextCommandDefault{\capitalmacron{@tabacckludge=}}~~
~~\DeclareTextCommandDefault{\capitalnewtie{\t}}~~
~~\DeclareTextCommandDefault{\capitalogonek{\k}}~~
~~\DeclareTextCommandDefault{\capitalring{\r}}~~
~~\DeclareTextCommandDefault{\capitaltie{\t}}~~
~~\DeclareTextCommandDefault{\capitaltilde{\~}}~~
~~\DeclareTextCommandDefault{\newtie{\t}}~~
209 \fi

The next two symbols exist in some fonts (faked?), but we ignore that to keep the subsets reasonable compact and most important linear.

```

210 \DeclareTextCommandDefault{\textlbrackdbl}
211 {\tc@check@symbol2\textlbrackdbl}
212 \DeclareTextCommandDefault{\textrbrackdbl}
213 {\tc@check@symbol2\textrbrackdbl}

```

Old style numerals are again in some fonts but using -0sF, etc. is the better approach to get them, so we claim they aren't in sub-encoding 2 as that's true for most fonts.

```

214 \DeclareTextCommandDefault{\texteightoldstyle}
215 {\tc@check@symbol2\texteightoldstyle}
216 \DeclareTextCommandDefault{\textfiveoldstyle}
217 {\tc@check@symbol2\textfiveoldstyle}
218 \DeclareTextCommandDefault{\textfouroldstyle}
219 {\tc@check@symbol2\textfouroldstyle}
220 \DeclareTextCommandDefault{\textnineoldstyle}
221 {\tc@check@symbol2\textnineoldstyle}
222 \DeclareTextCommandDefault{\textoneoldstyle}
223 {\tc@check@symbol2\textoneoldstyle}
224 \DeclareTextCommandDefault{\textsevenoldstyle}
225 {\tc@check@symbol2\textsevenoldstyle}
226 \DeclareTextCommandDefault{\textsixoldstyle}
227 {\tc@check@symbol2\textsixoldstyle}
228 \DeclareTextCommandDefault{\textthreeoldstyle}
229 {\tc@check@symbol2\textthreeoldstyle}
230 \DeclareTextCommandDefault{\texttwooldstyle}
231 {\tc@check@symbol2\texttwooldstyle}
232 \DeclareTextCommandDefault{\textzerooldstyle}
233 {\tc@check@symbol2\textzerooldstyle}

```

The next set of glyphs is special to TeX fonts (and available with a few older PS fonts supported through virtual fonts), but not any longer in the majority of fonts provided through autoinst, so we pretend there aren't available in sub-encoding 2 and below.

```

234 \DeclareTextCommandDefault{\textacutedbl}
235 {\tc@check@symbol2\textacutedbl}
236 \DeclareTextCommandDefault{\textasciiacute}
237 {\tc@check@symbol2\textasciiacute}
238 \DeclareTextCommandDefault{\textasciibreve}
239 {\tc@check@symbol2\textasciibreve}
240 \DeclareTextCommandDefault{\textasciicaron}
241 {\tc@check@symbol2\textasciicaron}
242 \DeclareTextCommandDefault{\textasciidieresis}
243 {\tc@check@symbol2\textasciidieresis}
244 \DeclareTextCommandDefault{\textasciigrave}
245 {\tc@check@symbol2\textasciigrave}
246 \DeclareTextCommandDefault{\textasciimacron}
247 {\tc@check@symbol2\textasciimacron}
248 \DeclareTextCommandDefault{\textgravedbl}
249 {\tc@check@symbol2\textgravedbl}
250 \DeclareTextCommandDefault{\texttildelow}
251 {\tc@check@symbol2\texttildelow}

```

Finally those below are only available in CM-based fonts but in no font that has its origin outside of the TeX world.

```

252 \DeclareTextCommandDefault{\textbaht}
253 {\tc@check@symbol2\textbaht}

```

```

254 \DeclareTextCommandDefault{\textbigcircle}
255 {\textsymbol{tc@check}{symbol2}{textbigcircle}}
256 \DeclareTextCommandDefault{\textborn}
257 {\textsymbol{tc@check}{symbol2}{textborn}}
258 \DeclareTextCommandDefault{\textcentoldstyle}
259 {\textsymbol{tc@check}{symbol2}{textcentoldstyle}}
260 \DeclareTextCommandDefault{\textcircledP}
261 {\textsymbol{tc@check}{symbol2}{textcircledP}}
262 \DeclareTextCommandDefault{\textcopyleft}
263 {\textsymbol{tc@check}{symbol2}{textcopyleft}}
264 \DeclareTextCommandDefault{\textdblhyphenchar}
265 {\textsymbol{tc@check}{symbol2}{textdblhyphenchar}}
266 \DeclareTextCommandDefault{\textdblhyphen}
267 {\textsymbol{tc@check}{symbol2}{textdblhyphen}}
268 \DeclareTextCommandDefault{\textdied}
269 {\textsymbol{tc@check}{symbol2}{textdied}}
270 \DeclareTextCommandDefault{\textdiscount}
271 {\textsymbol{tc@check}{symbol2}{textdiscount}}
272 \DeclareTextCommandDefault{\textdivorced}
273 {\textsymbol{tc@check}{symbol2}{textdivorced}}
274 \DeclareTextCommandDefault{\textdollaroldstyle}
275 {\textsymbol{tc@check}{symbol2}{textdollaroldstyle}}
276 \DeclareTextCommandDefault{\textguarani}
277 {\textsymbol{tc@check}{symbol2}{textguarani}}
278 \DeclareTextCommandDefault{\textleaf}
279 {\textsymbol{tc@check}{symbol2}{textleaf}}
280 \DeclareTextCommandDefault{\textlquill}
281 {\textsymbol{tc@check}{symbol2}{textlquill}}
282 \DeclareTextCommandDefault{\textmarried}
283 {\textsymbol{tc@check}{symbol2}{textmarried}}
284 \DeclareTextCommandDefault{\textmho}
285 {\textsymbol{tc@check}{symbol2}{textmho}}
286 \DeclareTextCommandDefault{\textmusicalnote}
287 {\textsymbol{tc@check}{symbol2}{textmusicalnote}}
288 \DeclareTextCommandDefault{\textnaira}
289 {\textsymbol{tc@check}{symbol2}{textnaira}}
290 \DeclareTextCommandDefault{\textopenbullet}
291 {\textsymbol{tc@check}{symbol2}{textopenbullet}}
292 \DeclareTextCommandDefault{\textpeso}
293 {\textsymbol{tc@check}{symbol2}{textpeso}}
294 \DeclareTextCommandDefault{\textpilcrow}
295 {\textsymbol{tc@check}{symbol2}{textpilcrow}}
296 \DeclareTextCommandDefault{\textrecipe}
297 {\textsymbol{tc@check}{symbol2}{textrecipe}}
298 \DeclareTextCommandDefault{\textremark}
299 {\textsymbol{tc@check}{symbol2}{textremark}}
300 \DeclareTextCommandDefault{\textrquill}
301 {\textsymbol{tc@check}{symbol2}{textrquill}}
302 \DeclareTextCommandDefault{\textservicemark}
303 {\textsymbol{tc@check}{symbol2}{textservicemark}}
304 \DeclareTextCommandDefault{\textsurd}
305 {\textsymbol{tc@check}{symbol2}{textsurd}}

```

The `\textpertenthousand` also belongs in this group but here we have a choice: in T1 there is a definition for `\textpertenthousand` making the symbol up from % and

`\char 24` (twice) but in many fonts that char doesn't exist and the slot is reused for random ligatures. So better not use it because often it is wrong. But pointing to TS1 is also not great as only a few fonts have it as a real symbol, so we get a substitution to CM or LM.

Alternatively we could just state that the symbol is unavailable in those fonts. For now I substitute.

```
306 \DeclareTextCommandDefault{\textpertenthousand}
307 {\tc@check@symbol2\textpertenthousand}
308 \UndeclareTextCommand{\textpertenthousand}{T1}
```

### 1.3 Unavailable in sub-encoding 3 and higher

Sub-encoding 2 is the one where we loose many symbols. In the higher-numbered sub-encodings we see only a few dropped additionally.

```
309 \DeclareTextCommandDefault{\textlangle}
310 {\tc@check@symbol3\textlangle}
311 \DeclareTextCommandDefault{\textrangle}
312 {\tc@check@symbol3\textrangle}
```

### 1.4 Unavailable in sub-encoding 4 and higher

```
313 \DeclareTextCommandDefault{\textcolonmonetary}
314 {\tc@check@symbol4\textcolonmonetary}
315 \DeclareTextCommandDefault{\textdong}
316 {\tc@check@symbol4\textdong}
317 \DeclareTextCommandDefault{\textdownarrow}
318 {\tc@check@symbol4\textdownarrow}
319 \DeclareTextCommandDefault{\textleftarrow}
320 {\tc@check@symbol4\textleftarrow}
321 \DeclareTextCommandDefault{\textlira}
322 {\tc@check@symbol4\textlira}
323 \DeclareTextCommandDefault{\textrightarrow}
324 {\tc@check@symbol4\textrightarrow}
325 \DeclareTextCommandDefault{\textuparrow}
326 {\tc@check@symbol4\textuparrow}
327 \DeclareTextCommandDefault{\textwon}
328 {\tc@check@symbol4\textwon}
```

### 1.5 Unavailable in sub-encoding 5 (most older PS fonts) and higher

Most older PS fonts (supported in TeX since the early nineties when virtual fonts became available) are sorted under this sub-encoding. But in reality, many of them don't have all glyphs that should be available in sub-encoding 5. Instead they show little squares, i.e., they produce “tofu” if you are unlucky.

But the coverage is so random that it is impossible to sort them properly and if we tried to ensure that they only typeset those glyphs that are really always available, we would have to put them all into sub-encoding 9; so putting them into 5 is really a compromise.

Modern fonts usually don't typeset a tofu character if a glyph is missing. They are therefore only classified as sub-encoding 5 if they really support its glyph set completely.

```
329 \DeclareTextCommandDefault{\textestimated}
330 {\tc@check@symbol5\textestimated}
```

```

331 \DeclareTextCommandDefault{\textnumero}
332 {\tc@check@symbol5\textnumero}

```

## 1.6 Unavailable in sub-encoding 6 and higher

```

333 \DeclareTextCommandDefault{\textflorin}
334 {\tc@check@symbol6\textflorin}
335 \DeclareTextCommandDefault{\textcurrency}
336 {\tc@check@symbol6\textcurrency}

```

## 1.7 Unavailable in sub-encoding 7 and higher

```

337 \DeclareTextCommandDefault{\textfractionsolidus}
338 {\tc@check@symbol7\textfractionsolidus}
339 \DeclareTextCommandDefault{\textohm}
340 {\tc@check@symbol7\textohm}
341 \DeclareTextCommandDefault{\textmu}
342 {\tc@check@symbol7\textmu}
343 \DeclareTextCommandDefault{\textminus}
344 {\tc@check@symbol7\textminus}

```

## 1.8 Unavailable in sub-encoding 8 and higher

```

345 \DeclareTextCommandDefault{\textblank}
346 {\tc@check@symbol{8}\textblank}
347 \DeclareTextCommandDefault{\textinterrobangdown}
348 {\tc@check@symbol{8}\textinterrobangdown}
349 \DeclareTextCommandDefault{\textinterrobang}
350 {\tc@check@symbol{8}\textinterrobang}

```

Fonts with this sub-encoding don't have a Euro symbol, but instead of substituting we fake it.

```

351 \DeclareTextCommandDefault{\texteuro}
352 {\CheckEncodingSubset\UseTextSymbol{TS1}\tc@fake@euro{8}\texteuro}

```

## 1.9 Unavailable in Sub-encoding 9 (most missing)

```

353 \DeclareTextCommandDefault{\textcelsius}
354 {\tc@check@symbol{9}\textcelsius}
355 \DeclareTextCommandDefault{\textonesuperior}
356 {\tc@check@symbol{9}\textonesuperior}
357 \DeclareTextCommandDefault{\textthreequartersemdash}
358 {\tc@check@symbol{9}\textthreequartersemdash}
359 \DeclareTextCommandDefault{\textthreesuperior}
360 {\tc@check@symbol{9}\textthreesuperior}
361 \DeclareTextCommandDefault{\texttwelveudash}
362 {\tc@check@symbol{9}\texttwelveudash}
363 \DeclareTextCommandDefault{\texttwosuperior}
364 {\tc@check@symbol{9}\texttwosuperior}
365 \DeclareTextCommandDefault{\textbardbl}
366 {\tc@check@symbol{9}\textbardbl}

```

## 2 Unicode engine specials

If we are using a unicode engine we handle some glyphs differently, so this here are the definitions for the Unicode encoding (overwriting the defaults above).

```

367 \ifx \Umathcode\@undefined \else

```

This set should be taken from TS1 encoding even if it means you get it from the default font for that encoding.

```

368 \%DeclareTextSymbol{\textcopyleft}{TS1}{171}
369 \%DeclareTextSymbol{\textdblhyphen}{TS1}{45}
370 \%DeclareTextSymbol{\textdblhyphenchar}{TS1}{127}
371 \%DeclareTextSymbol{\textquotestraightbase}{TS1}{13}
372 \%DeclareTextSymbol{\textquotestraightdblbase}{TS1}{18}
373 \%DeclareTextSymbol{\textleaf}{TS1}{108}
374 \%DeclareTextSymbol{\texttwelveudash}{TS1}{21}
375 \%DeclareTextSymbol{\textthreequartersemdash}{TS1}{22}

```

If oldstyle numerals are asked for we just use \oldstylenums.

```

376 \DeclareTextCommand{\textzerooldstyle} \UnicodeEncodingName{\oldstylenums{0}}
377 \DeclareTextCommand{\textoneoldstyle} \UnicodeEncodingName{\oldstylenums{1}}
378 \DeclareTextCommand{\texttwooldstyle} \UnicodeEncodingName{\oldstylenums{2}}
379 \DeclareTextCommand{\textthreeoldstyle} \UnicodeEncodingName{\oldstylenums{3}}
380 \DeclareTextCommand{\textfouroldstyle} \UnicodeEncodingName{\oldstylenums{4}}
381 \DeclareTextCommand{\textfiveoldstyle} \UnicodeEncodingName{\oldstylenums{5}}
382 \DeclareTextCommand{\textsixoldstyle} \UnicodeEncodingName{\oldstylenums{6}}
383 \DeclareTextCommand{\textsevenoldstyle} \UnicodeEncodingName{\oldstylenums{7}}
384 \DeclareTextCommand{\texteightoldstyle} \UnicodeEncodingName{\oldstylenums{8}}
385 \DeclareTextCommand{\textnineoldstyle} \UnicodeEncodingName{\oldstylenums{9}}

```

These have Unicode slots so this should be integrated into TU explicitly

```

386 \DeclareTextSymbol{\textpilcrow} \UnicodeEncodingName{"00B6}
387 \DeclareTextSymbol{\textborn} \UnicodeEncodingName{"002A}
388 \DeclareTextSymbol{\textdied} \UnicodeEncodingName{"2020}
389 \DeclareTextSymbol{\textlbrackdbl} \UnicodeEncodingName{"27E6}
390 \DeclareTextSymbol{\textrbrackdbl} \UnicodeEncodingName{"27E7}
391 \DeclareTextSymbol{\textguarani} \UnicodeEncodingName{"20B2}

```

We could make \textcentoldstyle and \textdollaroldstyle point to dollar and cent in the Unicode encoding

```

392 \%DeclareTextSymbol{\textcentoldstyle} \UnicodeEncodingName{"00A2}
393 \%DeclareTextSymbol{\textdollaroldstyle} \UnicodeEncodingName{"0024}

```

but I think it is better to pick them up from TS1 even if that usually means LMR fonts

```

394 \DeclareTextSymbol{\textdollaroldstyle}{TS1}{138}
395 \DeclareTextSymbol{\textcentoldstyle} {TS1}{139}
396 \fi % --- END of Unicode engines specials

```

### 3 Font family sub-encodings setup

We declare the subsets for a good number of fonts in the kernel ...

But first the default for anything that is not declared. We use 9 which is most likely much too conservative, but with the advantage that we aren't getting missing glyphs (or at least that this is very unlikely). For nearly all font in the TeX Live distribution of 2019 "correct" classifications are given below, so that this default is only used for new font families, and over time the right classifications can be added here too.

```

397 \DeclareEncodingSubset{TS1}{?}{9}

```

This first block contains the fonts that have been already supported by the `textcomp` package way back, i.e., the font families that have TeX support since the mid-nineties.

```

398 \DeclareEncodingSubset{TS1}{ccr} {0}
399 \DeclareEncodingSubset{TS1}{cmbr} {0}
400 \DeclareEncodingSubset{TS1}{cmr} {0}
401 \DeclareEncodingSubset{TS1}{cmss} {0}
402 \DeclareEncodingSubset{TS1}{cmtl} {0}
403 \DeclareEncodingSubset{TS1}{cmtt} {0}
404 \DeclareEncodingSubset{TS1}{cmvtt} {0}
405 \DeclareEncodingSubset{TS1}{pxr} {0}
406 \DeclareEncodingSubset{TS1}{pxss} {0}
407 \DeclareEncodingSubset{TS1}{pxtt} {0}
408 \DeclareEncodingSubset{TS1}{qag} {0}
409 \DeclareEncodingSubset{TS1}{qbk} {0}
410 \DeclareEncodingSubset{TS1}{qcr} {0}
411 \DeclareEncodingSubset{TS1}{qcs} {0}
412 \DeclareEncodingSubset{TS1}{qhvc} {0}
413 \DeclareEncodingSubset{TS1}{qhv} {0}
414 \DeclareEncodingSubset{TS1}{qp1} {0}
415 \DeclareEncodingSubset{TS1}{qtm} {0}
416 \DeclareEncodingSubset{TS1}{qzc} {0}
417 \DeclareEncodingSubset{TS1}{txr} {0}
418 \DeclareEncodingSubset{TS1}{txss} {0}
419 \DeclareEncodingSubset{TS1}{txtt} {0}

420 \DeclareEncodingSubset{TS1}{lmr} {1}
421 \DeclareEncodingSubset{TS1}{lmdh} {1}
422 \DeclareEncodingSubset{TS1}{lmss} {1}
423 \DeclareEncodingSubset{TS1}{lmssq} {1}
424 \DeclareEncodingSubset{TS1}{lmvtt} {1}
425 \DeclareEncodingSubset{TS1}{lmtt} {1} % missing TM, SM and
 % pertenthousand for some reason
426

427 \DeclareEncodingSubset{TS1}{ptmx} {2}
428 \DeclareEncodingSubset{TS1}{ptmj} {2}
429 \DeclareEncodingSubset{TS1}{ul8} {2}

430 \DeclareEncodingSubset{TS1}{bch} {5} % tofu for blank, ohm
431 \DeclareEncodingSubset{TS1}{futj} {5} % tofu for blank, interrobang/down, ohm
432 \DeclareEncodingSubset{TS1}{futs} {5} % tofu for blank, ohm
433 \DeclareEncodingSubset{TS1}{futx} {5} % probably (currently broken distrib)
434 \DeclareEncodingSubset{TS1}{pag} {5} % tofu for blank, interrobang/down, ohm
435 \DeclareEncodingSubset{TS1}{pbk} {5} % tofu for blank, interrobang/down, ohm
436 \DeclareEncodingSubset{TS1}{pcr} {5} % tofu for blank, interrobang/down, ohm
437 \DeclareEncodingSubset{TS1}{phv} {5} % tofu for blank, interrobang/down, ohm
438 \DeclareEncodingSubset{TS1}{pnc} {5} % tofu for blank, interrobang/down, ohm
439 \DeclareEncodingSubset{TS1}{pplj} {5} % tofu for blank
440 \DeclareEncodingSubset{TS1}{pplx} {5} % tofu for blank
441 \DeclareEncodingSubset{TS1}{ppl} {5} % tofu for blank interrobang/down
442 \DeclareEncodingSubset{TS1}{ptm} {5} % tofu for blank, interrobang/down, ohm
443 \DeclareEncodingSubset{TS1}{pzc} {5} % tofu for blank, interrobang/down, ohm
444 \DeclareEncodingSubset{TS1}{ul9} {5} % tofu for blank, interrobang/down, ohm

445 \DeclareEncodingSubset{TS1}{dayroms}{6} % tofu for blank, interrobang/down, ohm
446 \DeclareEncodingSubset{TS1}{dayrom}{6} % tofu for blank, interrobang/down, ohm
447 \DeclareEncodingSubset{TS1}{augie}{8} % really only missing euro

```

```

448 \DeclareEncodingSubset{TS1}{put} {8}
449 \DeclareEncodingSubset{TS1}{uag} {8} % probably (currently broken distrib)
450 \DeclareEncodingSubset{TS1}{ugq} {8}
451 \DeclareEncodingSubset{TS1}{zi4} {9}

```

LucidaBright (sold through TUG) probably not quite correct, I guess as I have the older fonts ...

```

452 \DeclareEncodingSubset{TS1}{hls} {5}
453 \DeclareEncodingSubset{TS1}{hlst} {5}
454 \DeclareEncodingSubset{TS1}{hlct} {5}
455 \DeclareEncodingSubset{TS1}{hh} {5}
456 \DeclareEncodingSubset{TS1}{hlx} {8}
457 \DeclareEncodingSubset{TS1}{hlce} {8}
458 \DeclareEncodingSubset{TS1}{hlcn} {8}
459 \DeclareEncodingSubset{TS1}{hlcw} {8}
460 \DeclareEncodingSubset{TS1}{hlcf} {8}

```

Below are the newer fonts that have support files for L<sup>A</sup>T<sub>E</sub>X. With very few exceptions the classifications are done so that all characters are correctly produced (either being available in the font or substituted).

There are a few fonts that contain “tofu” squares in places (instead of a real glyph) and in a few cases some really seldom needed chars are unavailable, i.e., produce missing glyphs (to avoid that a large number of available chars are unnecessarily substituted).

```

461 \DeclareEncodingSubset{TS1}{lato-*} {0} % with a bunch of tofu inside
462 \DeclareEncodingSubset{TS1}{opensans-*} {0} % with a bunch of tofu inside
463 \DeclareEncodingSubset{TS1}{cantarell-*} {0} % with a bunch of tofu inside
464 \DeclareEncodingSubset{TS1}{fb8-*} {0} % missing centoldstyle
465 \DeclareEncodingSubset{TS1}{tli} {1} % with lots of tofu inside
466 \DeclareEncodingSubset{TS1}{Alegreya-*} {2}
467 \DeclareEncodingSubset{TS1}{AlegreyaSans-*} {2}
468 \DeclareEncodingSubset{TS1}{DejaVuSans-TLF} {2}
469 \DeclareEncodingSubset{TS1}{DejaVuSansCondensed-TLF} {2}
470 \DeclareEncodingSubset{TS1}{DejaVuSansMono-TLF} {2}
471 \DeclareEncodingSubset{TS1}{EBGaramond-*} {2}
472 \DeclareEncodingSubset{TS1}{Tempora-TLF} {2}
473 \DeclareEncodingSubset{TS1}{Tempora-T0sF} {2}
474 \DeclareEncodingSubset{TS1}{Arimo-TLF} {3}
475 \DeclareEncodingSubset{TS1}{Carlito-*} {3}
476 \DeclareEncodingSubset{TS1}{FiraSans-*} {3}
477 \DeclareEncodingSubset{TS1}{IBMPlexSans-TLF} {3}
478 \DeclareEncodingSubset{TS1}{Merriweather-OsF} {3}
479 \DeclareEncodingSubset{TS1}{Montserrat-*} {3}
480 \DeclareEncodingSubset{TS1}{MontserratAlternates-*} {3}
481 \DeclareEncodingSubset{TS1}{SourceCodePro-TLF} {3}
482 \DeclareEncodingSubset{TS1}{SourceCodePro-T0sF} {3}
483 \DeclareEncodingSubset{TS1}{SourceSansPro-*} {3}
484 \DeclareEncodingSubset{TS1}{SourceSerifPro-*} {3}
485 \DeclareEncodingSubset{TS1}{Tinos-TLF} {3}
486 \DeclareEncodingSubset{TS1}{AccanthisADFSStdNoThree-LF} {4}
487 \DeclareEncodingSubset{TS1}{Cabin-TLF} {4}
488 \DeclareEncodingSubset{TS1}{Caladea-TLF} {4}
489 \DeclareEncodingSubset{TS1}{Chivo-*} {4}

```

```

490 \DeclareEncodingSubset{TS1}{ClearSans-TLF} {4}
491 \DeclareEncodingSubset{TS1}{Coelacanth-LF} {4}
492 \DeclareEncodingSubset{TS1}{CrimsonPro-*} {4}
493 \DeclareEncodingSubset{TS1}{FiraMono-TLF} {4}
494 \DeclareEncodingSubset{TS1}{FiraMono-T0sF} {4}
495 \DeclareEncodingSubset{TS1}{Go-TLF} {4}
496 \DeclareEncodingSubset{TS1}{GoMono-TLF} {4}
497 \DeclareEncodingSubset{TS1}{InriaSans-*} {4}
498 \DeclareEncodingSubset{TS1}{InriaSerif-*} {4}
499 \DeclareEncodingSubset{TS1}{LibertinusSans-*} {4}
500 \DeclareEncodingSubset{TS1}{LibertinusSerif-*} {4}
501 \DeclareEncodingSubset{TS1}{LibreBodoni-TLF} {4}
502 \DeclareEncodingSubset{TS1}{LibreFranklin-TLF} {4}
503 \DeclareEncodingSubset{TS1}{LinguisticsPro-LF} {4}
504 \DeclareEncodingSubset{TS1}{LinguisticsPro-OsF} {4}
505 \DeclareEncodingSubset{TS1}{LinuxBiolinumT-*} {4}
506 \DeclareEncodingSubset{TS1}{LinuxLibertineT-*} {4}
507 \DeclareEncodingSubset{TS1}{MerriweatherSans-OsF} {4}
508 \DeclareEncodingSubset{TS1}{MintSpirit-*} {4}
509 \DeclareEncodingSubset{TS1}{MintSpiritNoTwo-*} {4}
510 \DeclareEncodingSubset{TS1}{PTMono-TLF} {4}
511 \DeclareEncodingSubset{TS1}{PTSans-TLF} {4}
512 \DeclareEncodingSubset{TS1}{PTSansCaption-TLF} {4}
513 \DeclareEncodingSubset{TS1}{PTSansNarrow-TLF} {4}
514 \DeclareEncodingSubset{TS1}{PTSerif-TLF} {4}
515 \DeclareEncodingSubset{TS1}{PTSerifCaption-TLF} {4}
516 \DeclareEncodingSubset{TS1}{Raleway-TLF} {4}
517 \DeclareEncodingSubset{TS1}{Raleway-T0sF} {4}
518 \DeclareEncodingSubset{TS1}{Roboto-*} {4}
519 \DeclareEncodingSubset{TS1}{RobotoMono-TLF} {4}
520 \DeclareEncodingSubset{TS1}{RobotoSlab-TLF} {4}
521 \DeclareEncodingSubset{TS1}{Rosario-*} {4}
522 \DeclareEncodingSubset{TS1}{SticksTooText-*} {4}
523 \DeclareEncodingSubset{TS1}{UniversalisADFStd-LF} {4}

524 \DeclareEncodingSubset{TS1}{Almendra-OsF} {5}
525 \DeclareEncodingSubset{TS1}{Baskervaldx-*} {5}
526 \DeclareEncodingSubset{TS1}{BaskervilleF-*} {5}
527 \DeclareEncodingSubset{TS1}{Bitter-TLF} {5}
528 \DeclareEncodingSubset{TS1}{Cinzel-LF} {5}
529 \DeclareEncodingSubset{TS1}{CinzelDecorative-LF} {5}
530 \DeclareEncodingSubset{TS1}{DejaVuSerif-TLF} {5}
531 \DeclareEncodingSubset{TS1}{DejaVuSerifCondensed-TLF} {5}
532 \DeclareEncodingSubset{TS1}{GilliusADF-LF} {5}
533 \DeclareEncodingSubset{TS1}{GilliusADFCond-LF} {5}
534 \DeclareEncodingSubset{TS1}{GilliusADFNNoTwo-LF} {5}
535 \DeclareEncodingSubset{TS1}{GilliusADFNNoTwoCond-LF} {5}
536 \DeclareEncodingSubset{TS1}{LobsterTwo-LF} {5}
537 \DeclareEncodingSubset{TS1}{OldStandard-TLF} {5}
538 \DeclareEncodingSubset{TS1}{PlayfairDisplay-TLF} {5}
539 \DeclareEncodingSubset{TS1}{PlayfairDisplay-T0sF} {5}
540 \DeclareEncodingSubset{TS1}{TheanoDidot-TLF} {5}
541 \DeclareEncodingSubset{TS1}{TheanoDidot-T0sF} {5}
542 \DeclareEncodingSubset{TS1}{TheanoModern-TLF} {5}
543 \DeclareEncodingSubset{TS1}{TheanoModern-T0sF} {5}

```

```

544 \DeclareEncodingSubset{TS1}{TheanoOldStyle-TLF} {5}
545 \DeclareEncodingSubset{TS1}{TheanoOldStyle-T0sF} {5}
546 \DeclareEncodingSubset{TS1}{Crimson-TLF} {6}
547 \DeclareEncodingSubset{TS1}{IBMPlexMono-TLF} {6}
548 \DeclareEncodingSubset{TS1}{IBMPlexSerif-TLF} {6}
549 \DeclareEncodingSubset{TS1}{LibertinusMono-TLF} {6}
550 \DeclareEncodingSubset{TS1}{LibertinusSerifDisplay-LF}{6}
551 \DeclareEncodingSubset{TS1}{LinuxLibertineDisplayT-*} {6}
552 \DeclareEncodingSubset{TS1}{LinuxLibertineMonoT-LF} {6}
553 \DeclareEncodingSubset{TS1}{LinuxLibertineMonoT-TLF} {6}
554 \DeclareEncodingSubset{TS1}{Overlock-LF} {6}
555 \DeclareEncodingSubset{TS1}{CormorantGaramond-*} {7}
556 \DeclareEncodingSubset{TS1}{Heuristica-TLF} {7}
557 \DeclareEncodingSubset{TS1}{Heuristica-T0sF} {7}
558 \DeclareEncodingSubset{TS1}{IMFELLEnglish-TLF} {7}
559 \DeclareEncodingSubset{TS1}{LibreBaskerville-TLF} {7}
560 \DeclareEncodingSubset{TS1}{LibreCaslon-*} {7}
561 \DeclareEncodingSubset{TS1}{Marcellus-LF} {7}
562 \DeclareEncodingSubset{TS1}{NotoSans-*} {7}
563 \DeclareEncodingSubset{TS1}{NotoSansMono-TLF} {7}
564 \DeclareEncodingSubset{TS1}{NotoSansMono-T0sF} {7}
565 \DeclareEncodingSubset{TS1}{NotoSerif-*} {7}
566 \DeclareEncodingSubset{TS1}{Quattrocento-TLF} {7}
567 \DeclareEncodingSubset{TS1}{QuattrocentoSans-TLF} {7}
568 \DeclareEncodingSubset{TS1}{XCharter-TLF} {7}
569 \DeclareEncodingSubset{TS1}{XCharter-T0sF} {7}
570 \DeclareEncodingSubset{TS1}{erewhon-*} {7}
571 \DeclareEncodingSubset{TS1}{ComicNeue-TLF} {7}
572 \DeclareEncodingSubset{TS1}{ComicNeueAngular-TLF} {7}
573 \DeclareEncodingSubset{TS1}{Forum-LF} {7} % the superiors are missing
574 \DeclareEncodingSubset{TS1}{Cochineal-*} {8}
575 \DeclareEncodingSubset{TS1}{AlgolRevived-TLF} {9}

```

## 4 Legacy symbol support for lists and footnote symbols

```

\UseLegacyTextSymbols
576 \def\UseLegacyTextSymbols{%
577 \DeclareTextSymbolDefault{\textasteriskcentered}{OMS}%
578 \DeclareTextSymbolDefault{\textbardbl}{OMS}%
579 \DeclareTextSymbolDefault{\textbullet}{OMS}%
580 \DeclareTextSymbolDefault{\textdaggerdbl}{OMS}%
581 \DeclareTextSymbolDefault{\textdagger}{OMS}%
582 \DeclareTextSymbolDefault{\textparagraph}{OMS}%
583 \DeclareTextSymbolDefault{\textperiodcentered}{OMS}%
584 \DeclareTextSymbolDefault{\textsection}{OMS}%
585 \UndeclareTextCommand{\textsection}{T1}%
586 \expandafter\let\csname oldstylenums \expandafter\endcsname
587 \csname legacyoldstylenums \endcsname
588 }

```

(End definition for \UseLegacyTextSymbols.)

```
\textlegacyasteriskcentered
 \textlegacybardbl
 \textlegacybullet
\textlegacydaggerdbl
 \textlegacydagger
\textlegacyparagraph
\textlegacyperiodcentered
 \textlegacysection
```

Here are new names for the legacy symbols that L<sup>A</sup>T<sub>E</sub>X used to pick up from the OMS encoded fonts (and used for itemize lists or footnote symbols).

We go the roundabout way via separate OMS declarations so that

```
\renewcommand\textbullet{\textlegacybullet}
```

doesn't produce an endless loop.

```
589 \DeclareTextSymbol{\textlegacyasteriskcentered}{OMS}{3} % "03
590 \DeclareTextSymbol{\textlegacybardbl}{OMS}{107} % "6B
591 \DeclareTextSymbol{\textlegacybullet}{OMS}{15} % "0F
592 \DeclareTextSymbol{\textlegacydaggerdbl}{OMS}{122} % "7A
593 \DeclareTextSymbol{\textlegacydagger}{OMS}{121} % "79
594 \DeclareTextSymbol{\textlegacyparagraph}{OMS}{123} % "7B
595 \DeclareTextSymbol{\textlegacyperiodcentered}{OMS}{1} % "01
596 \DeclareTextSymbol{\textlegacysection}{OMS}{120} % "78

597 \DeclareTextSymbolDefault{\textlegacyasteriskcentered}{OMS}
598 \DeclareTextSymbolDefault{\textlegacybardbl}{OMS}
599 \DeclareTextSymbolDefault{\textlegacybullet}{OMS}
600 \DeclareTextSymbolDefault{\textlegacydaggerdbl}{OMS}
601 \DeclareTextSymbolDefault{\textlegacydagger}{OMS}
602 \DeclareTextSymbolDefault{\textlegacyparagraph}{OMS}
603 \DeclareTextSymbolDefault{\textlegacyperiodcentered}{OMS}
604 \DeclareTextSymbolDefault{\textlegacysection}{OMS}
```

(End definition for \textlegacyasteriskcentered and others.)

Supporting rollback ...

```
605 </2ekernel | latexrelease>
606 <latexrelease>
607 <latexrelease>\IncludeInRelease{0000/00/00}%
608 <latexrelease> {lttextcomp}{Undefine text companion symbols}%
609 <latexrelease>
610 <latexrelease>\DeclareRobustCommand\oldstylenums[1]{%
611 <latexrelease> \begingroup
612 <latexrelease> \spaceskip\fontdimen\tw@\font
613 <latexrelease> \usefont{OML}{\rmdefault}{\f@series}{it}%
614 <latexrelease> \mathgroup\symletters #1%
615 <latexrelease> \endgroup
616 <latexrelease>}%
617 <latexrelease>\let\legacyoldstylenums\@undefined
618 <latexrelease>\def\textcompsubstdefault{cmr}%
619 <latexrelease>
620 <latexrelease>\let\DeclareEncodingSubset\@undefined
621 <latexrelease>\let\CheckEncodingSubset\@undefined
622 <latexrelease>
623 <latexrelease>\DeclareTextSymbolDefault{\textdollar}{OT1}%
624 <latexrelease>\DeclareTextSymbolDefault{\textsterling}{OT1}%
625 <latexrelease>\DeclareTextCommand{\textdollar}{OT1}{\hmode@bgroup
626 <latexrelease> \ifdim \fontdimen\@ne\font >\z@
627 <latexrelease> \slshape
628 <latexrelease> \else
629 <latexrelease> \upshape
630 <latexrelease> \fi
```

```

631 〈\latexrelease〉 \char`\'$\\egroup}
632 〈\latexrelease〉\DeclareTextCommand{\textsterling}{\OT1}{\hmode@bgroup
633 〈\latexrelease〉 \ifdim \fontdimen\@ne\font >\z@
634 〈\latexrelease〉 \itshape
635 〈\latexrelease〉 \else
636 〈\latexrelease〉 \fontshape{ui}\selectfont
637 〈\latexrelease〉 \fi
638 〈\latexrelease〉 \char`\'$\\egroup}
639 〈\latexrelease〉\DeclareTextCommand{\textperthousand}{T1}
640 〈\latexrelease〉 {\%\char 24 }
641 〈\latexrelease〉
642 〈\latexrelease〉\DeclareTextSymbolDefault{\textasteriskcentered}{OMS}
643 〈\latexrelease〉\DeclareTextSymbolDefault{\textbullet}{OMS}
644 〈\latexrelease〉\DeclareTextSymbolDefault{\textdaggerdbl}{OMS}
645 〈\latexrelease〉\DeclareTextSymbolDefault{\textdagger}{OMS}
646 〈\latexrelease〉\DeclareTextSymbolDefault{\textparagraph}{OMS}
647 〈\latexrelease〉\DeclareTextSymbolDefault{\textperiodcentered}{OMS}
648 〈\latexrelease〉\DeclareTextSymbolDefault{\textsection}{OMS}
649 〈\latexrelease〉
650 〈\latexrelease〉\DeclareTextSymbolDefault{\textbardbl}{OMS}
651 〈\latexrelease〉\let\textbrokenbar\@undefined
652 〈\latexrelease〉\let\textcelsius\@undefined
653 〈\latexrelease〉\let\textcent\@undefined
654 〈\latexrelease〉\DeclareTextCommandDefault{\textcopyright}
655 〈\latexrelease〉 \{\textcircled{c}\}
656 〈\latexrelease〉\let\textdegree\@undefined
657 〈\latexrelease〉\let\textdiv\@undefined
658 〈\latexrelease〉\let\textlnot\@undefined
659 〈\latexrelease〉\let\textonehalf\@undefined
660 〈\latexrelease〉\let\textonequarter\@undefined
661 〈\latexrelease〉\let\textonesuperior\@undefined
662 〈\latexrelease〉\DeclareTextCommandDefault{\textordfeminine}
663 〈\latexrelease〉 \{a\}
664 〈\latexrelease〉\DeclareTextCommandDefault{\textordmasculine}
665 〈\latexrelease〉 \{o\}
666 〈\latexrelease〉\let\textpm\@undefined
667 〈\latexrelease〉\let\textquotesingle\@undefined
668 〈\latexrelease〉\let\textquotestraightbase\@undefined
669 〈\latexrelease〉\let\textquotestraightdblbase\@undefined
670 〈\latexrelease〉\DeclareTextCommandDefault{\textregistered}
671 〈\latexrelease〉 \{\textcircled{r}\}
672 〈\latexrelease〉 \check@mathfonts\fontsize\sf@size\z@
673 〈\latexrelease〉 \math@fontsfalse\selectfont R\}
674 〈\latexrelease〉\let\textthreequartersemdash\@undefined
675 〈\latexrelease〉\let\textthreequarters\@undefined
676 〈\latexrelease〉\let\textthreesuperior\@undefined
677 〈\latexrelease〉\let\texttimes\@undefined
678 〈\latexrelease〉\DeclareTextCommandDefault{\texttrademark}
679 〈\latexrelease〉 \{TM\}
680 〈\latexrelease〉\let\texttwelveudash\@undefined
681 〈\latexrelease〉\let\texttwosuperior\@undefined
682 〈\latexrelease〉\let\textyen\@undefined
683 〈\latexrelease〉
684 〈\latexrelease〉\let\textcapitalcompwordmark\@undefined

```

```

685 〈\latexrelease〉\let\textascendercompwordmark\@undefined
686 〈\latexrelease〉
687 〈\latexrelease〉\DeclareTextAccentDefault{\textcircled}{OMS}
688 〈\latexrelease〉\DeclareTextAccentDefault{\t}{OML}
689 〈\latexrelease〉
690 〈\latexrelease〉\let\capitalacute\@undefined
691 〈\latexrelease〉\let\capitalbreve\@undefined
692 〈\latexrelease〉\let\capitalcaron\@undefined
693 〈\latexrelease〉\let\capitalcedilla\@undefined
694 〈\latexrelease〉\let\capitalcircumflex\@undefined
695 〈\latexrelease〉\let\capitaldieresis\@undefined
696 〈\latexrelease〉\let\capitaldotaccent\@undefined
697 〈\latexrelease〉\let\capitalgrave\@undefined
698 〈\latexrelease〉\let\capitalhungarumlaut\@undefined
699 〈\latexrelease〉\let\capitalmacron\@undefined
700 〈\latexrelease〉\let\capitalnewtie\@undefined
701 〈\latexrelease〉\let\capitalogonek\@undefined
702 〈\latexrelease〉\let\capitalring\@undefined
703 〈\latexrelease〉\let\capitaltie\@undefined
704 〈\latexrelease〉\let\capitaltilde\@undefined
705 〈\latexrelease〉\let\newtie\@undefined
706 〈\latexrelease〉
707 〈\latexrelease〉\let\textlbrackdbl\@undefined
708 〈\latexrelease〉\let\textrbrackdbl\@undefined
709 〈\latexrelease〉
710 〈\latexrelease〉\let\texteightoldstyle\@undefined
711 〈\latexrelease〉\let\textfiveoldstyle\@undefined
712 〈\latexrelease〉\let\textfouroldstyle\@undefined
713 〈\latexrelease〉\let\textnineoldstyle\@undefined
714 〈\latexrelease〉\let\textoneoldstyle\@undefined
715 〈\latexrelease〉\let\textsevenoldstyle\@undefined
716 〈\latexrelease〉\let\textsixoldstyle\@undefined
717 〈\latexrelease〉\let\textthreeoldstyle\@undefined
718 〈\latexrelease〉\let\texttwooldstyle\@undefined
719 〈\latexrelease〉\let\textzerooldstyle\@undefined
720 〈\latexrelease〉
721 〈\latexrelease〉\let\textacutedbl\@undefined
722 〈\latexrelease〉\let\textasciacute\@undefined
723 〈\latexrelease〉\let\textasciibreve\@undefined
724 〈\latexrelease〉\let\textasciicaron\@undefined
725 〈\latexrelease〉\let\textasciidieresis\@undefined
726 〈\latexrelease〉\let\textasciigrave\@undefined
727 〈\latexrelease〉\let\textasciimacron\@undefined
728 〈\latexrelease〉\let\textgravedbl\@undefined
729 〈\latexrelease〉\let\texttildelow\@undefined
730 〈\latexrelease〉
731 〈\latexrelease〉\let\textbaht\@undefined
732 〈\latexrelease〉\let\textbigcircle\@undefined
733 〈\latexrelease〉\let\textborn\@undefined
734 〈\latexrelease〉\let\textcentoldstyle\@undefined
735 〈\latexrelease〉\let\textcircledP\@undefined
736 〈\latexrelease〉\let\textcopyleft\@undefined
737 〈\latexrelease〉\let\textdblhyphenchar\@undefined
738 〈\latexrelease〉\let\textdblhyphen\@undefined

```

```

739 〈\latexrelease〉\let\textdied\@undefined
740 〈\latexrelease〉\let\textdiscount\@undefined
741 〈\latexrelease〉\let\textdivorced\@undefined
742 〈\latexrelease〉\let\textdollaroldstyle\@undefined
743 〈\latexrelease〉\let\textguarani\@undefined
744 〈\latexrelease〉\let\textleaf\@undefined
745 〈\latexrelease〉\let\textlquill\@undefined
746 〈\latexrelease〉\let\textmarried\@undefined
747 〈\latexrelease〉\let\textmho\@undefined
748 〈\latexrelease〉\let\textmusicalnote\@undefined
749 〈\latexrelease〉\let\textnaira\@undefined
750 〈\latexrelease〉\let\textopenbullet\@undefined
751 〈\latexrelease〉\let\textpeso\@undefined
752 〈\latexrelease〉\let\textpilcrow\@undefined
753 〈\latexrelease〉\let\textrecipe\@undefined
754 〈\latexrelease〉\let\textreferencemark\@undefined
755 〈\latexrelease〉\let\textrquill\@undefined
756 〈\latexrelease〉\let\textservicemark\@undefined
757 〈\latexrelease〉\let\textsurd\@undefined
758 〈\latexrelease〉
759 〈\latexrelease〉\DeclareTextCommand{\textpertenthousand}{T1}
760 〈\latexrelease〉 { \%\char 24\char 24 }
761 〈\latexrelease〉
762 〈\latexrelease〉\let\textlangle\@undefined
763 〈\latexrelease〉\let\texttriangle\@undefined
764 〈\latexrelease〉
765 〈\latexrelease〉\let\textcolonmonetary\@undefined
766 〈\latexrelease〉\let\textdong\@undefined
767 〈\latexrelease〉\let\textdownarrow\@undefined
768 〈\latexrelease〉\let\textleftarrow\@undefined
769 〈\latexrelease〉\let\textlira\@undefined
770 〈\latexrelease〉\let\textrightarrow\@undefined
771 〈\latexrelease〉\let\textuparrow\@undefined
772 〈\latexrelease〉\let\textwon\@undefined
773 〈\latexrelease〉
774 〈\latexrelease〉\let\textestimated\@undefined
775 〈\latexrelease〉\let\textnumero\@undefined
776 〈\latexrelease〉
777 〈\latexrelease〉\let\textflorin\@undefined
778 〈\latexrelease〉\let\textcurrency\@undefined
779 〈\latexrelease〉
780 〈\latexrelease〉\let\textfractionsolidus\@undefined
781 〈\latexrelease〉\let\textohm\@undefined
782 〈\latexrelease〉\let\textmu\@undefined
783 〈\latexrelease〉\let\textminus\@undefined
784 〈\latexrelease〉
785 〈\latexrelease〉\let\textblank\@undefined
786 〈\latexrelease〉\let\textinterrobangdown\@undefined
787 〈\latexrelease〉\let\textinterrobang\@undefined
788 〈\latexrelease〉
789 〈\latexrelease〉\let\texteuro\@undefined
790 〈\latexrelease〉
791 〈\latexrelease〉\let\textcelsius\@undefined
792 〈\latexrelease〉\let\textonesuperior\@undefined

```

```

793 〈\latexrelease〉\let\textthreequartersemdash\@undefined
794 〈\latexrelease〉\let\textthreesuperior\@undefined
795 〈\latexrelease〉\let\texttwelveudash\@undefined
796 〈\latexrelease〉\let\texttwosuperior\@undefined
797 〈\latexrelease〉\let\textbardbl\@undefined
798 〈\latexrelease〉
799 〈\latexrelease〉\let\UseLegacyTextSymbols\@undefined
800 〈\latexrelease〉\let\textlegacyasteriskcentered\@undefined
801 〈\latexrelease〉\let\textlegacybardbl\@undefined
802 〈\latexrelease〉\let\textlegacybullet\@undefined
803 〈\latexrelease〉\let\textlegacydaggerdbl\@undefined
804 〈\latexrelease〉\let\textlegacydagger\@undefined
805 〈\latexrelease〉\let\textlegacyparagraph\@undefined
806 〈\latexrelease〉\let\textlegacyperiodcentered\@undefined
807 〈\latexrelease〉\let\textlegacysection\@undefined
808 〈\latexrelease〉
809 〈\latexrelease〉\EndModuleRelease

```

## 5 The `textcomp` package

```

810 〈*TS1sty〉
811 〈\providecommand\DeclareRelease[3]{}〉
812 〈\providecommand\DeclareCurrentRelease[2]{}〉
813
814 〈\DeclareRelease{}{2018-08-11}{textcomp-2018-08-11.sty}〉
815 〈\DeclareCurrentRelease{}{2020-02-02}〉
816
817 〈\ProvidesPackage{textcomp}〉
818 〔2020/02/02 v2.0n Standard LaTeX package〕

```

A precaution in case this is used without rebuilding the format.

```
819 〈\NeedsTeXFormat{LaTeX2e}〔2020/02/02〕〉
```

This is implemented by defining the default subset:

```

820 〈\DeclareOption{full}{\DeclareEncodingSubset{TS1}{?}{0}}〉
821 〈\DeclareOption{almostfull}{\DeclareEncodingSubset{TS1}{?}{1}}〉
822 〈\DeclareOption{euro}{\DeclareEncodingSubset{TS1}{?}{8}}〉
823 〈\DeclareOption{safe}{\DeclareEncodingSubset{TS1}{?}{9}}〉

```

The default is set up in the kernel is “safe” these days for unknown fonts but LaTeX has definitions for most families so it seldom applies.

If a different default is used then one needs to check the results to ensure that there aren’t “missing glyphs”.

The next set of options define the warning level (default in the kernel is info only). Using the package options you can change this behavior.

```

824 〈\DeclareOption{error}{\gdef\tc@errorwarn{\PackageError{textcomp}}}
825 〈\gdef\tc@errorwarn{\PackageError{textcomp}}〉
826 〈\DeclareOption{warn}{\gdef\tc@errorwarn{\PackageWarning{textcomp}}}
827 〈\gdef\tc@errorwarn{\PackageWarning{textcomp}}〉
828 〈\DeclareOption{info}{\gdef\tc@errorwarn{\PackageInfo{textcomp}}}
829 〈\gdef\tc@errorwarn{\PackageInfo{textcomp}}〉
830 〈\DeclareOption{quiet}{\gdef\tc@errorwarn{\gdef\tc@errorwarn{}}〉

```

The “force” option basically changes the sub-encoding to that of the default (which, unless changes, is 9 these days), i.e., it no longer depends on the font in use. This is

mainly there because it might have been used in older documents, but not something that is recommended.

```

831 \DeclareOption{force}{%
832 \def\CheckEncodingSubset#1#2#3#4#5{%
833 \ifnum #4>%
834 \csname #2:\endcsname
835 \relax
836 \expandafter\@firstoftwo
837 \else
838 \expandafter\@secondoftwo
839 \fi
840 {#1{#2}}{#3}%
841 #5}%
842 }
843 \ExecuteOptions{info}
844 \ProcessOptions\relax

```

There is not much else to do nowadays, because everything is already set up in the L<sup>A</sup>T<sub>E</sub>X kernel.

```

845 \InputIfFileExists{textcomp.cfg}
846 {\PackageInfo{textcomp}{Local configuration file used}}{}%
847 </TS1sty>

```

## 5.1 The old textcomp package code

This section contains the old code for the textcomp package and its documentation. It is only used if we roll back prior to 2020. Thus all the rest is mainly for historians. Note that the old code categorized in the sub-encodings only into 6 classes not 10.

```

848 <*TS1oldsty>
849 \ProvidesPackage{textcomp}
850 [2018/08/11 v2.0j Standard LATEX package]

```

This one is for the TS1 encoding which contains text symbols for use with the T1-encoded text fonts. It therefore first inputs the file `TS1enc.def` and then sets (or resets) the defaults for the symbols it contains. The result of this is that when one of these symbols is accessed and the current encoding does not provide it, the symbol will be supplied by a silent, local change to this encoding.

Since many PostScript fonts only implement a subset of TS1 many commands only produce black blobs of ink. To resolve the resulting problems a number of options have been introduced and some code has been developed to distinguish sub-encodings.

The sub-encodings have a numerical id and are defined as follows for TS1:

**#5** those TS1 symbols that are also in the ISO-Adobe character set; without `\textcurrency`, which is often misused for the Euro. Older Type1 fonts from the non-T<sub>E</sub>X world provide only this subset.

**#4** = #5 + `\texteuro`. Most newer fonts provide this.

**#3** = #4 + `\textomega`. Can also be described as  $TS1 \cap (ISO-Adobe \cup MacRoman)$ . (Except for the missing "currency".)

**#2** = #3 + `\textestimated` + `\textcurrency`. Can also be described as  $TS1 \cap Adobe-Western-2$ . This may be relevant for OpenType fonts, which usually show the Adobe-Western-2 character set.

**#1** = TS1 without `\textcircled` and `\t`. These two glyphs are often not implemented and if their kernel defaults are changed commands like `\copyright` unnecessarily fail.

**#0** = full TS1

And here a summary to go in the transcript file:

```

851 \PackageInfo{textcomp}{Sub-encoding information:\MessageBreak
852 \space\space 5 = only ISO-Adobe without
853 \string\textcurrency\MessageBreak
854 \space\space 4 = 5 + \string\texteuro\MessageBreak
855 \space\space 3 = 4 + \string\textohm\MessageBreak
856 \space\space 2 = 3 + \noexpand\textestimated+
857 \string\textcurrency\MessageBreak
858 \space\space 1 = TS1 - \noexpand\textcircled-
859 \string\t\MessageBreak
860 \space\space 0 = TS1 (full)\MessageBreak
861 Font families with sub-encoding setting implement\MessageBreak
862 only a restricted character set as indicated.\MessageBreak
863 Family '?' is the default used for unknown fonts.\MessageBreak
864 See the documentation for details@gobble}

```

`\DeclareEncodingSubset` An encoding subset to which a font family belongs is declared by the command `\DeclareEncodingSubset` that takes the major encoding as the first argument (e.g., TS1), the family name as the second argument (e.g., cmr), and the subset encoding id as a third, (e.g., 0 for cmr).

The default encoding subset to use when nothing is known about the current font family is named ?.

```

865 \def\DeclareEncodingSubset#1#2#3{%
866 \@ifundefined{#1:#2}{%
867 {\PackageInfo{textcomp}{Setting #2 sub-encoding to #1/#3}}%
868 {\PackageInfo{textcomp}{Changing #2 sub-encoding to #1/#3}}%
869 }{\@namedef{#1:#2}{#3}}%
870 }@\onlypreamble\DeclareEncodingSubset

```

(End definition for `\DeclareEncodingSubset`.)

The options for the package are the following:

**safe** for unknown font families enables only symbols that are also in the ISO-Adobe character set; without "currency", which is often misused for the Euro. Older Type1 fonts from the non-TeX world provide only this subset.

**euro** enables the "safe" symbols plus the `\texteuro` command. Most newer fonts provide this.

**full** enables all TS1 commands; useful only with fonts like EC or CM bright.

**almostfull** same as "full", except that `\textcircled` and `\t` are *not* redefined from their defaults to avoid that commands like `\copyright` suddenly no longer work.

**force** ignore all subset encoding definitions stored in the package itself or in the configuration file and always use the default subset as specified by one of the other options (seldom useful, only dangerous).

\iftc@forced Switch used to implement the **force** option  
871 \newif\iftc@forced \tc@forcedfalse

(End definition for \iftc@forced.)

This is implemented by defining the default subset:

872 \DeclareOption{full}{\DeclareEncodingSubset{TS1}{?}{0}}  
873 \DeclareOption{almostfull}{\DeclareEncodingSubset{TS1}{?}{1}}  
874 \DeclareOption{euro}{\DeclareEncodingSubset{TS1}{?}{4}}  
875 \DeclareOption{safe}{\DeclareEncodingSubset{TS1}{?}{5}}

The default is “almostfull” which means that old documents will work except that \textcircled and \t will use the kernel defaults (with the advantage that this also works if the current font (as often the case) doesn’t implement these glyphs.

The “force” option simply sets the switch to true.

876 \DeclareOption{force}{\tc@forcedtrue}

The suggestions to user is to use the “safe” option always unless that balks in which case they could switch to “almostfull” but then better check their output manually.

877 \def\tc@errorwarn{\PackageError}  
878 \DeclareOption{warn}{\gdef\tc@errorwarn#1#2#3{\PackageWarning{#1}{#2}}}  
879 \DeclareOption{quiet}{\gdef\tc@errorwarn#1#2#3{}}  
880 \ExecuteOptions{almostfull}  
881 \ProcessOptions\relax

\CheckEncodingSubset The command \CheckEncodingSubset will check if the current font family has the right encoding subset to typeset a certain command. It takes five arguments as follows: first argument is either \UseTextSymbol, \UseTextAccent depending on whether or not the symbol is a text symbol or a text accent.

The second argument is the encoding from which this symbol should be fetched.

The third argument is either a fake accessor command or an error message. the code in that argument (if ever executed) receives two arguments: #2 and #5 of \CheckEncodingSubset.

Argument four is the subset encoding id to test against: if this value is higher than the subset id of the current font family then we typeset the symbol, i.e., execute #1{#2}#5 otherwise it runs #3#5, e.g., to produce an error message or fake the glyph somehow.

Argument five is the symbol or accent command that is being checked.

For usage examples see definitions below.

882 \iftc@forced

If the “force” option was given we always use the default for testing against.

883 \def\CheckEncodingSubset#1#2#3#4#5{  
884 \ifnum #4>%  
885 \csname #2:\endcsname  
886 \relax  
887 \expandafter\@firstoftwo  
888 \else  
889 \expandafter\@secondoftwo  
890 \fi  
891 {#1{#2}}{#3}%  
892 #5%  
893 }

In normal circumstances the test is a bit more complicated: first check if there exists a macro `\langle arg2\rangle:\langle current-family\rangle` and if so use that value to test against, otherwise use the default to test against.

```

894 \else
895 \def\CheckEncodingSubset#1#2#3#4#5{%
896 \ifnum #4>%
897 \expandafter\ifx\csname #2:\f@family\endcsname\relax
898 \csname #2:?\endcsname
899 \else
900 \csname #2:\f@family\endcsname
901 \fi
902 \relax
903 \expandafter\@firstoftwo
904 \else
905 \expandafter\@secondoftwo
906 \fi
907 {#1{#2}}{#3}%
908 #5%
909 }
910 \fi

```

(End definition for `\CheckEncodingSubset`.)

`\tc@subst`

```

911 \def\tc@subst#1{%
912 \tc@errorwarn{textcomp}%
913 {Symbol \string#1 not provided by\MessageBreak
914 font family \f@family\space
915 in TS1 encoding.\MessageBreak Default family used instead}\@eha
916 \bgroup\fontfamily\textcompsubstdefault\selectfont#1\egroup
917 }

```

(End definition for `\tc@subst`.)

`\tc@error` `\tc@error` is going to be used in arg #3 of `\CheckEncodingSubset` when a symbol is not available in a certain font family. It gets pass the encoding it normally lives in (arg one) and the name of the symbol or accent that has a problem.

```

918 % error commands take argument:
919 % #1 symbol to be used
920 \def\tc@error#1{%
921 \PackageError{textcomp}%
922 {Accent \string#1 not provided by\MessageBreak
923 font family \f@family\space
924 in TS1 encoding}\@eha
925 }

```

(End definition for `\tc@error`.)

`\tc@fake@euro` `\tc@fake@euro` is an example of a “fake” definition to use in arg #3 of `\CheckEncodingSubset` when a symbol is not available in a certain font family. Here we produce an Euro symbol by combining a “C” with a “=“.

```

926 \def\tc@fake@euro#1{%
927 \leavevmode
928 \PackageInfo{textcomp}{Faking \noexpand#1 for font family}

```

```

929 \f@family\MessageBreak in TS1 encoding}%
930 \valign{\##\cr
931 \vfil\hbox to 0.07em{\dimen@\f@size\p@
932 \math@fontsfalse
933 \fontsize{.7\dimen@}{\z@\selectfont=\hss}%
934 \vfil\cr%
935 \hbox{C}\crcr
936 }%
937 }

```

*(End definition for \tc@fake@euro.)*

\tc@check@symbol These are two abbreviations that we use below to check symbols and accents in TS1.  
\tc@check@accent Only there to save some space, e.g., we can then write

```
DeclareTextCommandDefault{\textcurrency}{\tc@check@symbol3\textcurrency}
```

to ensure that \textcurrency is only typeset if the current font has a TS1 subset id of less than 3. Otherwise \tc@error is called telling the user that for this font family \textcurrency is not available.

```

938 \def\tc@check@symbol{\CheckEncodingSubset\UseTextSymbol{TS1}\tc@subst}
939 \def\tc@check@accent{\CheckEncodingSubset\UseTextAccent{TS1}\tc@error}

```

*(End definition for \tc@check@symbol and \tc@check@accent.)*

We start with the commands that are “safe” and which can be unconditionally set up, first the accents...

```

940 \DeclareTextAccentDefault{\capitalcedilla}{TS1}
941 \DeclareTextAccentDefault{\capitalogonek}{TS1}
942 \DeclareTextAccentDefault{\capitalgrave}{TS1}
943 \DeclareTextAccentDefault{\capitalacute}{TS1}
944 \DeclareTextAccentDefault{\capitalcircumflex}{TS1}
945 \DeclareTextAccentDefault{\capitaltilde}{TS1}
946 \DeclareTextAccentDefault{\capitaldieresis}{TS1}
947 \DeclareTextAccentDefault{\capitalhungarumlaut}{TS1}
948 \DeclareTextAccentDefault{\capitalring}{TS1}
949 \DeclareTextAccentDefault{\capitalcaron}{TS1}
950 \DeclareTextAccentDefault{\capitalbreve}{TS1}
951 \DeclareTextAccentDefault{\capitalmacron}{TS1}
952 \DeclareTextAccentDefault{\capitaldotaccent}{TS1}

```

... and then the other glyphs.

```

953 \DeclareTextSymbolDefault{\textcapitalcompwordmark}{TS1}
954 \DeclareTextSymbolDefault{\textascendercompwordmark}{TS1}
955 \DeclareTextSymbolDefault{\textquotestraightbase}{TS1}
956 \DeclareTextSymbolDefault{\textquotestraightdblbase}{TS1}
957 \DeclareTextSymbolDefault{\texttwelveudash}{TS1}
958 \DeclareTextSymbolDefault{\textthreequartersemdash}{TS1}
959 \DeclareTextSymbolDefault{\textdollar}{TS1}
960 \DeclareTextSymbolDefault{\textquotesingle}{TS1}
961 \DeclareTextSymbolDefault{\textasteriskcentered}{TS1}
962 \DeclareTextSymbolDefault{\textfractionsolidus}{TS1}
963 \DeclareTextSymbolDefault{\textminus}{TS1}
964 \DeclareTextSymbolDefault{\textlbrackdbl}{TS1}
965 \DeclareTextSymbolDefault{\textrbrackdbl}{TS1}
966 \DeclareTextSymbolDefault{\textasciigrave}{TS1}

```

```

967 \DeclareTextSymbolDefault{\texttildelow}{TS1}
968 \DeclareTextSymbolDefault{\textasciibreve}{TS1}
969 \DeclareTextSymbolDefault{\textasciicaron}{TS1}
970 \DeclareTextSymbolDefault{\textgravedbl}{TS1}
971 \DeclareTextSymbolDefault{\textacutedbl}{TS1}
972 \DeclareTextSymbolDefault{\textdagger}{TS1}
973 \DeclareTextSymbolDefault{\textdaggerdbl}{TS1}
974 \DeclareTextSymbolDefault{\textbardbl}{TS1}
975 \DeclareTextSymbolDefault{\textperthousand}{TS1}
976 \DeclareTextSymbolDefault{\textbullet}{TS1}
977 \DeclareTextSymbolDefault{\textcelsius}{TS1}
978 \DeclareTextSymbolDefault{\textflorin}{TS1}
979 \DeclareTextSymbolDefault{\texttrademark}{TS1}
980 \DeclareTextSymbolDefault{\textcent}{TS1}
981 \DeclareTextSymbolDefault{\textsterling}{TS1}
982 \DeclareTextSymbolDefault{\textyen}{TS1}
983 \DeclareTextSymbolDefault{\textbrokenbar}{TS1}
984 \DeclareTextSymbolDefault{\textsection}{TS1}
985 \DeclareTextSymbolDefault{\textasciidieresis}{TS1}
986 \DeclareTextSymbolDefault{\textcopyright}{TS1}
987 \DeclareTextSymbolDefault{\textordfeminine}{TS1}
988 \DeclareTextSymbolDefault{\textlnot}{TS1}
989 \DeclareTextSymbolDefault{\textregistered}{TS1}
990 \DeclareTextSymbolDefault{\textasciimacron}{TS1}
991 \DeclareTextSymbolDefault{\textdegree}{TS1}
992 \DeclareTextSymbolDefault{\textpm}{TS1}
993 \DeclareTextSymbolDefault{\texttwosuperior}{TS1}
994 \DeclareTextSymbolDefault{\textthreesuperior}{TS1}
995 \DeclareTextSymbolDefault{\textasciacute}{TS1}
996 \DeclareTextSymbolDefault{\textmu}{TS1}
997 \DeclareTextSymbolDefault{\textparagraph}{TS1}
998 \DeclareTextSymbolDefault{\textperiodcentered}{TS1}
999 \DeclareTextSymbolDefault{\textonesuperior}{TS1}
1000 \DeclareTextSymbolDefault{\textordmasculine}{TS1}
1001 \DeclareTextSymbolDefault{\textonequarter}{TS1}
1002 \DeclareTextSymbolDefault{\textonehalf}{TS1}
1003 \DeclareTextSymbolDefault{\textthreequarters}{TS1}
1004 \DeclareTextSymbolDefault{\texttimes}{TS1}
1005 \DeclareTextSymbolDefault{\textdiv}{TS1}

```

The `\texteuro` is only available for subsets with id 4 or less. Otherwise we fake the glyph using `\tc@fake@euro`

```

1006 \DeclareTextCommandDefault{\texteuro}{%
1007 \CheckEncodingSubset\UseTextSymbol{TS1}\tc@fake@euro5\texteuro}

```

The `\textohm` is only available for subsets with id 3 or less. Otherwise we produce an error.

```

1008 \DeclareTextCommandDefault{\textohm}{\tc@check@symbol4\textohm}

```

The `\textestimated` and `\textcurrency` are only provided for fonts with subset encoding with id 2 or less.

```

1009 \DeclareTextCommandDefault{\textestimated}{%
1010 \tc@check@symbol3\textestimated}
1011 \DeclareTextCommandDefault{\textcurrency}{%
1012 \tc@check@symbol3\textcurrency}

```

Nearly all of the remaining glyphs are provided only with fonts with id 1 or 0, i.e., are essentially complete.

```
1013 \DeclareTextCommandDefault{\capitaltie}{%
1014 {\tc@check@accent2\capitaltie}}
1015 \DeclareTextCommandDefault{\newtie}{%
1016 {\tc@check@accent2\newtie}}
1017 \DeclareTextCommandDefault{\capitalnewtie}{%
1018 {\tc@check@accent2\capitalnewtie}}
1019 \DeclareTextCommandDefault{\textleftarrow}{%
1020 {\tc@check@symbol2\textleftarrow}}
1021 \DeclareTextCommandDefault{\textrightarrow}{%
1022 {\tc@check@symbol2\textrightarrow}}
1023 \DeclareTextCommandDefault{\textblank}{%
1024 {\tc@check@symbol2\textblank}}
1025 \DeclareTextCommandDefault{\textdblhyphen}{%
1026 {\tc@check@symbol2\textdblhyphen}}
1027 \DeclareTextCommandDefault{\textzerooldstyle}{%
1028 {\tc@check@symbol2\textzerooldstyle}}
1029 \DeclareTextCommandDefault{\textoneoldstyle}{%
1030 {\tc@check@symbol2\textoneoldstyle}}
1031 \DeclareTextCommandDefault{\texttwooldstyle}{%
1032 {\tc@check@symbol2\texttwooldstyle}}
1033 \DeclareTextCommandDefault{\textthreeoldstyle}{%
1034 {\tc@check@symbol2\textthreeoldstyle}}
1035 \DeclareTextCommandDefault{\textfouroldstyle}{%
1036 {\tc@check@symbol2\textfouroldstyle}}
1037 \DeclareTextCommandDefault{\textfiveoldstyle}{%
1038 {\tc@check@symbol2\textfiveoldstyle}}
1039 \DeclareTextCommandDefault{\textsixoldstyle}{%
1040 {\tc@check@symbol2\textsixoldstyle}}
1041 \DeclareTextCommandDefault{\textsevenoldstyle}{%
1042 {\tc@check@symbol2\textsevenoldstyle}}
1043 \DeclareTextCommandDefault{\texteightoldstyle}{%
1044 {\tc@check@symbol2\texteightoldstyle}}
1045 \DeclareTextCommandDefault{\textnineoldstyle}{%
1046 {\tc@check@symbol2\textnineoldstyle}}
1047 \DeclareTextCommandDefault{\textlangle}{%
1048 {\tc@check@symbol2\textlangle}}
1049 \DeclareTextCommandDefault{\textrangle}{%
1050 {\tc@check@symbol2\textrangle}}
1051 \DeclareTextCommandDefault{\textmho}{%
1052 {\tc@check@symbol2\textmho}}
1053 \DeclareTextCommandDefault{\textbigcircle}{%
1054 {\tc@check@symbol2\textbigcircle}}
1055 \DeclareTextCommandDefault{\textuparrow}{%
1056 {\tc@check@symbol2\textuparrow}}
1057 \DeclareTextCommandDefault{\textdownarrow}{%
1058 {\tc@check@symbol2\textdownarrow}}
1059 \DeclareTextCommandDefault{\textborn}{%
1060 {\tc@check@symbol2\textborn}}
1061 \DeclareTextCommandDefault{\textdivorced}{%
1062 {\tc@check@symbol2\textdivorced}}
1063 \DeclareTextCommandDefault{\textdied}{%
1064 {\tc@check@symbol2\textdied}}
```

```

1065 \DeclareTextCommandDefault{\textleaf}{%
1066 {\tc@check@symbol2\textleaf}%
1067 \DeclareTextCommandDefault{\textmarried}{%
1068 {\tc@check@symbol2\textmarried}%
1069 \DeclareTextCommandDefault{\textmusicalnote}{%
1070 {\tc@check@symbol2\textmusicalnote}%
1071 \DeclareTextCommandDefault{\textdblhyphenchar}{%
1072 {\tc@check@symbol2\textdblhyphenchar}%
1073 \DeclareTextCommandDefault{\textdollaroldstyle}{%
1074 {\tc@check@symbol2\textdollaroldstyle}%
1075 \DeclareTextCommandDefault{\textcentoldstyle}{%
1076 {\tc@check@symbol2\textcentoldstyle}%
1077 \DeclareTextCommandDefault{\textcolonmonetary}{%
1078 {\tc@check@symbol2\textcolonmonetary}%
1079 \DeclareTextCommandDefault{\textwon}{%
1080 {\tc@check@symbol2\textwon}%
1081 \DeclareTextCommandDefault{\textnaira}{%
1082 {\tc@check@symbol2\textnaira}%
1083 \DeclareTextCommandDefault{\textguarani}{%
1084 {\tc@check@symbol2\textguarani}%
1085 \DeclareTextCommandDefault{\textpeso}{%
1086 {\tc@check@symbol2\textpeso}%
1087 \DeclareTextCommandDefault{\textlira}{%
1088 {\tc@check@symbol2\textlira}%
1089 \DeclareTextCommandDefault{\textrecipe}{%
1090 {\tc@check@symbol2\textrecipe}%
1091 \DeclareTextCommandDefault{\textinterrobang}{%
1092 {\tc@check@symbol2\textinterrobang}%
1093 \DeclareTextCommandDefault{\textinterrobangdown}{%
1094 {\tc@check@symbol2\textinterrobangdown}%
1095 \DeclareTextCommandDefault{\textdong}{%
1096 {\tc@check@symbol2\textdong}%
1097 \DeclareTextCommandDefault{\textpertenthousand}{%
1098 {\tc@check@symbol2\textpertenthousand}%
1099 \DeclareTextCommandDefault{\textpilcrow}{%
1100 {\tc@check@symbol2\textpilcrow}%
1101 \DeclareTextCommandDefault{\textbaht}{%
1102 {\tc@check@symbol2\textbaht}%
1103 \DeclareTextCommandDefault{\textnumero}{%
1104 {\tc@check@symbol2\textnumero}%
1105 \DeclareTextCommandDefault{\textdiscount}{%
1106 {\tc@check@symbol2\textdiscount}%
1107 \DeclareTextCommandDefault{\textopenbullet}{%
1108 {\tc@check@symbol2\textopenbullet}%
1109 \DeclareTextCommandDefault{\textservicemark}{%
1110 {\tc@check@symbol2\textservicemark}%
1111 \DeclareTextCommandDefault{\textlquill}{%
1112 {\tc@check@symbol2\textlquill}%
1113 \DeclareTextCommandDefault{\textrquill}{%
1114 {\tc@check@symbol2\textrquill}%
1115 \DeclareTextCommandDefault{\textcopyleft}{%
1116 {\tc@check@symbol2\textcopyleft}%
1117 \DeclareTextCommandDefault{\textcircledP}{%
1118 {\tc@check@symbol2\textcircledP}%

```

```

1119 \DeclareTextCommandDefault{\textreferencemark}{%
1120 {\tc@check@symbol2\textreferencemark}%
1121 \DeclareTextCommandDefault{\textsurd}{%
1122 {\tc@check@symbol2\textsurd}%

```

The `\textcircled` and `\t` are handled specially, unless the current font has a subset id of 0 (i.e. full TS1) we pick the symbols up from the math font encodings, i.e., the third argument to `\CheckEncodingSubset` uses `\UseTextAccent` to get them from there.

```

1123 \DeclareTextCommandDefault{\textcircled}{%
1124 {\CheckEncodingSubset\UseTextAccent{TS1}}%
1125 {\UseTextAccent{OMS}}1\textcircled}%
1126 \DeclareTextCommandDefault{\t}{%
1127 {\CheckEncodingSubset\UseTextAccent{TS1}}%
1128 {\UseTextAccent{OML}}1\t}%

```

Finally input the encoding-specific definitions for TS1 thus making the top-level definitions optimized for this encoding (and not for the default encoding).

```
1129 \input{ts1enc.def}
```

Now having the new glyphs available we also want to make sure that they are used. For most cases this will automatically happen but for some glyphs there are inferior definitions already known to L<sup>A</sup>T<sub>E</sub>X which will prevent the usage of the TS1 versions. So we better get rid of them:

```

1130 \UndeclareTextCommand{\textsterling}{OT1}
1131 \UndeclareTextCommand{\textdollar} {OT1}

```

Similar declarations should probably be made for other encodings like OT4 if they are in use.

```

1132 %\UndeclareTextCommand{\textsterling}{OT4}
1133 %\UndeclareTextCommand{\textdollar} {OT4}

```

From the T1 encoding there are two candidates for removal: `%_o` and `%_oo` since these are both constructed from `%` followed by a tiny ‘`_`’ rather than being a single glyph. The problem with this approach is that in PostScript fonts this small zero is usually not available resulting in `%_■` rather than `%_o` while the real glyph (at least for `\textperthousand`) is available in the PostScript version of TS1. So for the moment we compromise by removing the T1 declaration for `\textperthousand` but keeping the one for `\textpertenthousand`. This will have the effect that with Computer Modern fonts everything will come out (although `%_o` and `%_oo` are not taken from the same physical font) and with PostScript fonts `%_o` will come out correctly while `%_oo` will most likely look like `%_■` — which is probably an improvement over just getting a single ‘`■`’ to indicate a completely missing glyph, which would happen if we also ‘undeclared’ `\textpertenthousand`.

```

1134 \UndeclareTextCommand{\textperthousand}{T1}
1135 %\UndeclareTextCommand{\textpertenthousand}{T1}

```

### 5.1.1 Supporting oldstyle digits

```

1136 \DeclareRobustCommand\oldstylenums[1]{%
1137 \begingroup
1138 \ifmmode
1139 \mathgroup\symletters #1%
1140 \else
1141 \CheckEncodingSubset\use@text@encoding{TS1}%
1142 {\PackageWarning{textcomp}%

```

```

1143 {Oldstyle digits unavailable for
1144 family \f@family.\MessageBreak
1145 Lining digits used instead}}%
1146 \tw@{\#1}%
1147 \fi
1148 \endgroup
1149 }

```

### 5.1.2 Subset encoding defaults

For many font families commonly used in the  $\text{\TeX}$  world we provide the subset encoding data here. Users can add additional font families in the file `textcomp.cfg` if they own other fonts.

However, if the option “forced” was given then all subset encoding specifications are ignored, so there is no point in setting any of them up:

```

1150 \iftc@forced \else
 Computer modern based fonts (e.g., CM, CM-Bright, Concrete):
1151 \DeclareEncodingSubset{TS1}{cmr} {0}
1152 \DeclareEncodingSubset{TS1}{cmss} {0}
1153 \DeclareEncodingSubset{TS1}{cmtt} {0}
1154 \DeclareEncodingSubset{TS1}{cmvtt} {0}
1155 \DeclareEncodingSubset{TS1}{cmbr} {0}
1156 \DeclareEncodingSubset{TS1}{cmtl} {0}
1157 \DeclareEncodingSubset{TS1}{ccr} {0}

 PSNFSS fonts:
1158 \DeclareEncodingSubset{TS1}{ptm} {4}
1159 \DeclareEncodingSubset{TS1}{pcr} {4}
1160 \DeclareEncodingSubset{TS1}{phv} {4}
1161 \DeclareEncodingSubset{TS1}{ppl} {3}
1162 \DeclareEncodingSubset{TS1}{pag} {4}
1163 \DeclareEncodingSubset{TS1}{pbk} {4}
1164 \DeclareEncodingSubset{TS1}{pnc} {4}
1165 \DeclareEncodingSubset{TS1}{pzc} {4}
1166 \DeclareEncodingSubset{TS1}{bch} {4}
1167 \DeclareEncodingSubset{TS1}{put} {5}

 Other CTAN fonts (probably not complete):
1168 \DeclareEncodingSubset{TS1}{uag} {5}
1169 \DeclareEncodingSubset{TS1}{ugq} {5}
1170 \DeclareEncodingSubset{TS1}{u18} {4}
1171 \DeclareEncodingSubset{TS1}{u19} {4} % (LuxiSans, one day)
1172 \DeclareEncodingSubset{TS1}{augie} {5}
1173 \DeclareEncodingSubset{TS1}{dayrom} {3}
1174 \DeclareEncodingSubset{TS1}{dayroms} {3}
1175 \DeclareEncodingSubset{TS1}{pxr} {0}
1176 \DeclareEncodingSubset{TS1}{pxss} {0}
1177 \DeclareEncodingSubset{TS1}{pxtt} {0}
1178 \DeclareEncodingSubset{TS1}{txr} {0}
1179 \DeclareEncodingSubset{TS1}{txss} {0}
1180 \DeclareEncodingSubset{TS1}{txtt} {0}

 Latin Modern and TeX Gyre:
1181 \DeclareEncodingSubset{TS1}{lmr} {0}
1182 \DeclareEncodingSubset{TS1}{lmdh} {0}

```

```

1183 \DeclareEncodingSubset{TS1}{lmss} {0}
1184 \DeclareEncodingSubset{TS1}{lmssq} {0}
1185 \DeclareEncodingSubset{TS1}{lmvtt} {0}
1186 \DeclareEncodingSubset{TS1}{lmtt} {0}
1187 \DeclareEncodingSubset{TS1}{qhv} {0}
1188 \DeclareEncodingSubset{TS1}{qag} {0}
1189 \DeclareEncodingSubset{TS1}{qbk} {0}
1190 \DeclareEncodingSubset{TS1}{qcr} {0}
1191 \DeclareEncodingSubset{TS1}{qcs} {0}
1192 \DeclareEncodingSubset{TS1}{qpl} {0}
1193 \DeclareEncodingSubset{TS1}{qtm} {0}
1194 \DeclareEncodingSubset{TS1}{qzc} {0}
1195 \DeclareEncodingSubset{TS1}{qhvc} {0}

```

Fourier-GUTenberg:

```

1196 \DeclareEncodingSubset{TS1}{futs} {4}
1197 \DeclareEncodingSubset{TS1}{futx} {4}
1198 \DeclareEncodingSubset{TS1}{futj} {4}

```

Y&Y's Lucida Bright

```

1199 \DeclareEncodingSubset{TS1}{hlh} {3}
1200 \DeclareEncodingSubset{TS1}{hls} {3}
1201 \DeclareEncodingSubset{TS1}{hlst} {3}

```

The remaining settings for Lucida are conservative: the following fonts contain the `\textohm` character but not the `\texteuro`, i.e., belong to neither subset 4 nor subset 3. If you want to use the `\textohm` with these fonts copy these definition to `textcomp.cfg` and change the subset to 3. However in that case make sure that you do not use the `\texteuro`.

```

1202 \DeclareEncodingSubset{TS1}{hlct} {5}
1203 \DeclareEncodingSubset{TS1}{hlx} {5}
1204 \DeclareEncodingSubset{TS1}{hlce} {5}
1205 \DeclareEncodingSubset{TS1}{hlcn} {5}
1206 \DeclareEncodingSubset{TS1}{hlcw} {5}
1207 \DeclareEncodingSubset{TS1}{hlcf} {5}

```

Other commercial families...

```

1208 \DeclareEncodingSubset{TS1}{pplx} {3}
1209 \DeclareEncodingSubset{TS1}{pplj} {3}
1210 \DeclareEncodingSubset{TS1}{ptmx} {4}
1211 \DeclareEncodingSubset{TS1}{ptmj} {4}

```

If the file `textcomp.cfg` exists it will be loaded at this point. This allows to define further subset encodings for font families not covered by default.

```

1212 \InputIfFileExists{textcomp.cfg}
1213 {\PackageInfo{textcomp}{Local configuration file used}}{}
1214 \fi
1215 </TS1oldsty>

```

# File F

## ltpageno.dtx

### 1 Page Numbering

Page numbers are produced by a page counter, used just like any other counter. The only difference is that `\c@page` contains the number of the next page to be output (the one currently being produced), rather than one minus it. Thus, it is normally initialized to 1 rather than 0. `\c@page` is defined to be `\count0`, rather than a count assigned by `\newcount`.

`\pagenumbering` The user sets the page number style with the `\pagenumbering{<foo>}` command, which sets the page counter to 1 and defines `\thepage` to be `\foo`. For example, `\pagenumbering{roman}` causes pages to be numbered i, ii, etc.

```
1 {*2ekernel}
2 \message{page nos.,}
3 \countdef\c@page=0 \c@page=1
4 \def\cl@page{}
5 \def\pagenumbering#1{%
6 \global\c@page \cne \gdef\thepage{\csname \#1\endcsname
7 \c@page}}
8 {/2ekernel}
```

# File G

## ltxref.dtx

### 1 Cross Referencing

The user writes `\label{<foo>}` to define the following cross-references:

`\ref{<foo>}`: value of most recently incremented referenceable counter. in the current environment. (Chapter, section, theorem and enumeration counters are referenceable, footnote counters are not.)

`\pageref{<foo>}`: page number at which `\label{foo}` command appeared. where foo can be any string of characters not containing ‘\’, ‘{’ or ‘}’.

Note: The scope of the `\label` command is delimited by environments, so  
`\begin{theorem} \label{foo} ... \end{theorem} \label{bar}`  
defines `\ref{foo}` to be the theorem number and `\ref{bar}` to be the current section number.

Note: `\label` does the right thing in terms of spacing – i.e., leaving a space on both sides of it is equivalent to leaving a space on either side.

#### 1.1 Cross Referencing

*Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*

```
1 {*2ekernel}
2 \message{x-ref,}
```

This is implemented as follows. A referenceable counter CNT is incremented by the command `\refstepcounter{CNT}`, which sets `\@currentlabel == {CNT}\eval(\p@cnt\theCNT)`. The command `\label{FOO}` then writes the following on file `\auxout`:

```
\newlabel{FOO}{\eval(\@currentlabel)\eval(\thepage)}
```

```
\ref{FOO} ==
BEGIN
 if \r@foo undefined
 then @refundefined := G T
 ??
 Warning: 'reference foo on page ... undefined'
 else \car \eval(\r@FOO)\nil
fi
END
```

```
\pageref{foo} =
BEGIN
 if \r@foo undefined
 then @refundefined := G T
 ??
 Warning: 'reference foo on page ... undefined'
 else \cdr \eval(\r@FOO)\nil
fi
```

END

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

\labelformat

A reference via \ref produces by default the data associated with the corresponding \label command (typically a number); any additional formatting has to be provided by the user. If, for example, references to equations are always to be typeset as “equation (number)”, one has to code “equation (\ref{key})”. With \labelformat there is a possibility to generate such frills automatically without resorting to low-level coding. The command takes two arguments: the first is the name of a counter and the second is its representation when referenced. This means that for a successful usage, one has to know the counter name being used for generating the label, though in practice this should not pose a problem. The current counter number is picked up as an argument. Here are two examples:

```
\labelformat{section}{section~#1}
\labelformat{equation}{equation~(#1)}
```

\Ref

A side effect of using \labelformat is that, depending on the defined formatting, it becomes impossible to use \ref at the beginning of a sentence (if its replacement text starts with a lowercase letter). To overcome this problem we introduce the command \Ref that behave like \ref except that it uppercases the first token of the generated string.

To make \Ref work properly the very first token in the second argument of \labelformat has to be a simple ASCII or UTF-8 letter, otherwise the capitalization will fail or worse, you will end up with some error messages. If you actually need something more complicated in this place (e.g., an accented letter not written as a UTF-8 character) you have to explicitly surround it with braces, to identify the part that needs to be capitalized. For example, for figure references in the Hungarian language you might want to write \labelformat{figure}{f\'a}bra-\thefigure or use \labelformat{figure}{\'abra-\thefigure} which avoids the brace problem.

\G@refundefinedtrue  
  \@refundefined

This does not save on name-space (since \G@refundefinedfalse was never needed) but it does make the implementation of such one-way switches more consistent. The extra macro to make the change is used since this change appears several times.

**Note** despite its name, \G@refundefinedtrue does *not* correspond to an \if command, and there is no matching ...false. It would be more natural to call the command \G@refundefined (as inspection of the change log will reveal) but unfortunately such a change would break any package that had defined a \ref-like command that mimicked the definition of \ref, calling \G@refundefinedtrue. Inspection of the T<sub>E</sub>X archives revealed several such packages, and so this command has been named ...true so that the definition of \ref need not be changed, and the packages will work without change.

```
3 % \newif\ifG@refundefined
4 % \def\G@refundefinedtrue{\global\let\ifG@refundefined\iftrue}
5 % \def\G@refundefinedfalse{\global\let\ifG@refundefined\iffalse}
6 \def\G@refundefinedtrue{%
7 \gdef\@refundefined{%
8 \@latex@warning@no@line{There were undefined references}}}
9 \let\@refundefined\relax
```

(End definition for \G@refundefinedtrue and \@refundefined.)

\ref Referencing a \label. RmS 91/10/25: added a few extra \reset@font, as suggested by Bernd Raichle  
 \pageref RmS 92/08/14: made \ref and \pageref robust  
 RmS 93/09/08: Added setting of refundefined switch.

```

10 \def\@setref#1#2#3{%
11 \ifx#1\relax
12 \protect\G@refundefinedtrue
13 \nfss@text{\reset@font\bfseries ??}%
14 \@latex@warning{Reference '#3' on page \thepage \space
15 undefined}%
16 \else
17 \expandafter#2#1\null
18 \fi}
19 \def\ref#1{\expandafter\@setref\csname r@#1\endcsname\@firstoftwo{#1}}
20 \def\pageref#1{\expandafter\@setref\csname r@#1\endcsname
21 \@secondoftwo{#1}}

```

(End definition for \ref, \pageref, and \@setref.)

\newlabel This command will be written to the .aux file to pass label information from one run to another.

\@newl@bel The internal form of \newlabel and \bincite. Note that this macro does it's work inside a group. That way the local assignments it needs to do don't clutter the save stack. This prevents large documents with many labels to run out of save stack.

```

22 \def\@newl@bel#1#2#3{%
23 \@ifundefined{#1@#2}%
24 \relax
25 {\gdef \@multiplelabels {%
26 \@latex@warning{no@line{There were multiply-defined labels}}%
27 \@latex@warning{no@line{Label '#2' multiply defined}}%
28 }%
29 \def\newlabel{\@newl@bel r}%
30 \onlypreamble\@newl@bel

```

(End definition for \newlabel and \@newl@bel.)

\if@multiplelabels This is redefined to produce a warning if at least one label is defined more than once. It is executed by the \enddocument command.

```

31 \let \@multiplelabels \relax

```

(End definition for \if@multiplelabels and \@multiplelabels.)

\label The commands \label and \refstepcounter have been changed to allow \protect'ed commands to work properly. For example,

```
\def\thechapter{\protect\foo{\arabic{chapter}. \roman{section}}}
```

will cause a \label{bar} command to define \ref{bar} to expand to something like \foo{4.d}. Change made 20 Jul 88.

```

32 \def\label#1{\@bsphack
33 \protected@write\@auxout{}{%
34 {\string\newlabel{#1}{\@currentlabel}\{\thepage\}}%
35 }\@esphack}

```

```

(End definition for \label1.)

36 {/2ekernel}
37 {*2ekernel | latexrelease}
38 <latexrelease>\IncludeInRelease{2020/10/01}%
39 <latexrelease> {\refstepcounter}{Add \@currentcounter}%

```

\refstepcounter Step the counter and allow for labels to point to its current value.

```

40 \def\@currentcounter{}
41 \def\refstepcounter#1{\stepcounter{#1}%
42 \edef\@currentcounter{#1}%
43 \protected@edef\@currentlabel

```

By generating the second csname first the \p@... command can grab it as an argument which can be helpful for more complicated typesetting arrangements.

The trick is to ensure that \csname the#1\endcsname is turned into a single token before \p@... is expanded further. This way, if the \p@... command is a macro with one argument it will receive \the.... With the original kernel code (i.e., without the \expandafter) it will instead pick up \csname which would be disastrous.

Using \expandafter instead of braces delimiting the argument is better because, assuming that the \p@... command is not defined as a macro with one argument, the braces will stay and prohibit kerning that might otherwise happen between the glyphs generated by \the... and surrounding glyphs.

```

44 {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}%
45 }

```

(End definition for \refstepcounter.)

\labelformat A shortcut to set the \p@... macro for a counter. It will pick up the counter representation as an argument so that it can be specially formatted.

```

46 \def\labelformat#1{\expandafter\def\csname p@#1\endcsname##1}

```

(End definition for \labelformat.)

\Ref This macro expands the result of \ref and then uppercases the first token. Only useful if the label was generated via \labelformat and contains some lower case letter at its start. If the label starts with a complicated construct (e.g., an accented letter that is provided via a command, e.g., \"a instead of a UTF-8 character like ä) one has to surround everything that needs uppercasing in a brace group in the definition of \labelformat.<sup>31</sup>

```

47 \DeclareRobustCommand\Ref[1]{\protected@edef\@tempa{\ref{#1}}%
48 \expandafter\MakeUppercase\@tempa}

```

(End definition for \Ref.)

```

49 {/2ekernel | latexrelease}
50 <latexrelease>\EndIncludeInRelease
51 <latexrelease>\IncludeInRelease{2019/10/01}%
52 <latexrelease> {\refstepcounter}{Add \labelformat and \Ref}%
53 <latexrelease>\let\@currentcounter\@undefined
54 <latexrelease>\def\refstepcounter#1{\stepcounter{#1}%
55 <latexrelease> \protected@edef\@currentlabel

```

---

<sup>31</sup>There is one problem with this approach: the braces are kept in a normal \ref which might spoil kerning. Perhaps one day this needs redoing.

```

56 〈latexrelease〉 {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}%
57 〈latexrelease〉}
58 〈latexrelease〉\def\labelformat#1{\expandafter\def\csname p@#1\endcsname##1}
59 〈latexrelease〉\DeclareRobustCommand\Ref[1]{\protected@edef\@tempa{\ref{#1}}}
60 〈latexrelease〉 \expandafter\MakeUppercase\@tempa}
61 〈latexrelease〉\EndIncludeInRelease
62 〈latexrelease〉\IncludeInRelease{0000/00/00}%
63 〈latexrelease〉 {\refstepcounter}{Add \labelformat and \Ref}%
64 〈latexrelease〉
65 〈latexrelease〉\def\refstepcounter#1{\stepcounter{#1}}%
66 〈latexrelease〉 \protected@edef\@currentlabel
67 〈latexrelease〉 {\csname p@#1\endcsname\csname the#1\endcsname}%
68 〈latexrelease〉}
69 〈latexrelease〉\let\labelformat\@undefined
70 〈latexrelease〉\let\Ref\@undefined
71 〈latexrelease〉
72 〈latexrelease〉\EndIncludeInRelease
73 〈*2ekernel〉

```

\@currentlabel Default for \label commands that come before any environment.

```
74 \def\@currentlabel{}
```

*(End definition for \@currentlabel.)*

```
75 〈/2ekernel〉
```

# File H

## ltmiscen.dtx

### 1 Miscellaneous Environments

This section implements the basic environment mechanism, and also a few specific environments including `document`, The math environments and related commands, the ‘flushing’ environments, (`center`, `flushleft`, `flushright`), and `verbatim`.

```
1 <*2ekernel>
2 \message{environments,}
```

#### 1.1 Environments

`\begin{foo}` and `\end{foo}` are used to delimit environment `foo`.

`\begin{foo}` starts a group and calls `\foo` if it is defined, otherwise it does nothing.

`\end{foo}` checks to see that it matches the corresponding `\begin` and if so, it calls `\endfoo` and does an `\endgroup`. Otherwise, `\end{foo}` does nothing.

If `\end{foo}` needs to ignore blanks after it, then `\endfoo` should globally set the `@ignore` switch true with `\@ignoretrue` (this will automatically be global).

NOTE: `\@end` is defined to be the `\end` command of TeX82.

`\enddocument` is the user’s command for ending the manuscript file.

`\stop` is a panic button — to end TeX in the middle.

*Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*

```
\enddocument ==
BEGIN
 \@checkend{document} %% checks for unmatched \begin
 \clearpage
 \begingroup
 if @filesw = true
 then close file @mainaux
 if G@refundefined = true
 then LaTeX Warning: 'There are undefined references.' fi
 if @multiplelabels = true
 then LaTeX Warning:
 'One or more label(s) multiply defined.'
 else
 \@setckpt {ARG1}{ARG2} == null
 \newlabel{LABEL}{VAL} ==
 BEGIN
 \reserved@a == VAL
 if def(\reserved@a) = def(\r@LABEL)
 else @tempswa := true fi
 END
 \bibcite{LABEL}{VAL} == null
 BEGIN
 \reserved@a == VAL
 if def(\reserved@a) = def(\g@LABEL)
 else @tempswa := true fi

```

```

 END
@tempswa := false
make @ a letter
\input \jobname.AUX
if @tempswa = true
 then LaTeX Warning: 'Label may have changed.
 Rerun to get cross-references right.'
fi fi fi
\endgroup
finish up
END

\@writefile{EXT}{ENTRY} ==
if tf@EXT undefined
 else \write\tf@EXT{ENTRY}
fi

```

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

\@currenvir The name of the current environment. Initialized to `document` so that `\end{document}` works correctly.

```
3 \def\@currenvir{document}
```

(*End definition for \@currenvir.*)

```

\if@ignore
\@ignoretrue
\@ignorefalse
4 \def\@ignorefalse{\global\let\if@ignore\iffalse}
5 \def\@ignoretrue {\global\let\if@ignore\iftrue}
6 \@ignorefalse

```

(*End definition for \if@ignore, \@ignoretrue, and \@ignorefalse.*)

\ignorespacesafterend

```
7 \let\ignorespacesafterend\@ignoretrue
```

(*End definition for \ignorespacesafterend.*)

\end{document})

```

8 </2ekernel>
9 {*2ekernel | latexrelease}
10 <latexrelease>\IncludeInRelease{2020/10/01}%
11 <latexrelease> {\enddocument}{Use Hooks}%
12 \def\enddocument{%

```

The `\end{document}` hook is executed first. If necessary it can contain a `\clearpage` to output dangling floats first. In this position it can also contain something like `\end{foo}` so that the whole document effectively starts and ends with some special environment. However, this must be used with care, eg if two applications would use this without knowledge of each other the order of the environments will be wrong after all. `\AtEndDocument` is redefined at this point so that and such commands that get into the hook do not chase their tail...

```

13 \kernel@before\enddocument
14 \UseOneTimeHook{\enddocument}%
15 \kernel@after\enddocument

```

```

16 \@checkend{document}%
17 \clearpage
18 \UseOneTimeHook{enddocument/afterlastpage}%
19 \@kernel@after@enddocument@afterlastpage
20 \begingroup
21 \if@filesw
22 \immediate\closeout\mainaux
23 \let\@setckpt\gobbletwo
24 \let\@newl@bel\@testdef

```

The previous line is equiv to setting

```

\def\newlabel{\@testdef r}%
\def\bibcite{\@testdef b}%

```

We use `\@@input` to load the .aux file, so that it doesn't show up in the list of files produced by `\listfiles`.

```

25 \@tempswafalse
26 \makeatletter \@@input\jobname.aux
27 \fi
28 \UseOneTimeHook{enddocument/afteraux}%

```

Next hook is expect to contain only code for writing info messages on the terminal.

```

29 \UseOneTimeHook{enddocument/info}%
30 \endgroup
31 \UseOneTimeHook{enddocument/end}%
32 \deadcycles{z@\@@end}%

```

The public hooks used in `\enddocument`:

```

33 \NewHook{enddocument}
34 \NewHook{enddocument/afterlastpage}
35 \NewHook{enddocument/afteraux}
36 \NewHook{enddocument/info}
37 \NewHook{enddocument/end}

```

This is one of the few places where we already add data and rules to a hook already in the kernel.

```

38 \AddToHook{enddocument/info}[kernel/filelist]{\@dofilelist}
39 \AddToHook{enddocument/info}[kernel/warnings]{\@enddocument@kernel@warnings}
40 \DeclareHookRule{enddocument/info}{kernel/filelist}{before}{kernel/warnings}
(End definition for \enddocument.)

```

`\@enddocument@kernel@warnings`

```

41 \def\@enddocument@kernel@warnings{%

```

First we check for font size substitution bigger than `\fontsubfuzz`. The `\relax` is necessary because this is a macro not a register.

```

42 \ifdim \font@submax >\fontsubfuzz\relax

```

In case you wonder about the `\gobbletwo` inside the message below, this is a horrible hack to remove the tokens `\on@line`. that are added by `\font@warning` at the end.

```

43 \@font@warning{Size substitutions with differences\MessageBreak
44 up to \font@submax\space have occurred.\gobbletwo}%
45 \fi

```

The macro `\@defaultsubs` is initially `\relax` but gets redefined to produce a warning if there have been some default font substitutions.

```
46 \@defaultsubs
```

The macro `\@refundefined` is initially `\relax` but gets redefined to produce a warning if there are undefined refs.

```
47 \@refundefined
```

If a label is defined more than once, `\@tempswa` will always be true and thus produce a “Label(s) may ...” warning. But since a rerun will not solve that problem (unless one uses a package like `variorom` that generates labels on the fly), we suppress this message.

```
48 \if@filesw
49 \ifx \@multiplelabels \relax
50 \if@tempswa
51 \@latex@warning@no@line{Label(s) may have changed.
52 Rerun to get cross-references right}%
53 \fi
54 \else
55 \@multiplelabels
56 \fi
57 \ifx \@extra@page@added \relax
58 \@latex@warning@no@line{Temporary extra page added at the end.
59 Rerun to get it removed}%
60 \fi
61 \fi
62 }
```

We could think of adding a warning that nothing can be corrected while `\nofiles` is in force. In the past the warnings related to the `.aux` file are simply suppressed in this case.

```
63
```

$$\begin{aligned}
64 & \langle /2ekernel | latexrelease \rangle \\
65 & \langle latexrelease \rangle \EndIncludeInRelease \\
66 & \langle latexrelease \rangle \IncludeInRelease{0000/00/00} \% \\
67 & \langle latexrelease \rangle \{\enddocument\}\{Use Hooks\} \% \\
68 & \langle latexrelease \rangle \def\enddocument{\% \\
69 & \langle latexrelease \rangle \let\AtEndDocument\@firstofone \\
70 & \langle latexrelease \rangle \@enddocumenthook \\
71 & \langle latexrelease \rangle \@checkend{document} \% \\
72 & \langle latexrelease \rangle \clearpage \\
73 & \langle latexrelease \rangle \begingroup \\
74 & \langle latexrelease \rangle \if@filesw \\
75 & \langle latexrelease \rangle \immediate\closeout\mainaux \\
76 & \langle latexrelease \rangle \let\@setckpt\@gobbletwo \\
77 & \langle latexrelease \rangle \let\@newlbel\@testdef \\
78 & \langle latexrelease \rangle \@tempswafalse \\
79 & \langle latexrelease \rangle \makeatletter \@@input\jobname.aux \\
80 & \langle latexrelease \rangle \fi \\
81 & \langle latexrelease \rangle \@dofilelist \\
82 & \langle latexrelease \rangle \ifdim \font@submax > \fontsubfuzz \relax \\
83 & \langle latexrelease \rangle \quad \@font@warning{Size substitutions with differences\MessageBreak \\
84 & \langle latexrelease \rangle \quad up to \font@submax\space have occurred.\@gobbletwo} \% \\
\end{aligned}$$

```

85 <latexrelease> \fi
86 <latexrelease> \@defaultsubs
87 <latexrelease> \@refundefined
88 <latexrelease> \if@filesw
89 <latexrelease> \ifx \c@multiplelabels \relax
90 <latexrelease> \if@tempswa
91 <latexrelease> \c@latex@warning@no@line{Label(s) may have changed.
92 <latexrelease> Rerun to get cross-references right}%
93 <latexrelease> \fi
94 <latexrelease> \else
95 <latexrelease> \c@multiplelabels
96 <latexrelease> \fi
97 <latexrelease> \fi
98 <latexrelease> \endgroup
99 <latexrelease> \deadcycles\z@\c@end}
100 <latexrelease>
101 <latexrelease>\let\c@enddocument\kernel@c@warnings\c@undefined
102 <latexrelease>
103 <latexrelease>\EndIncludeInRelease
104 <*2ekernel>

```

\@kernel@before@cenddocument The \@kernel@before@cenddocument hook is slightly different because we initialize it with \par so that \cenddocument always returns to vertical mode as its first action.

```

105 </2ekernel>
106 <*2ekernel | latexrelease>
107 <latexrelease>\IncludeInRelease{2021/06/01}%
108 <latexrelease> {\@kernel@before@cenddocument}{kernel before hook}%
109 \def\@kernel@before@cenddocument{\par}
110 </2ekernel | latexrelease>
111 <latexrelease>\EndIncludeInRelease

```

The rollback code renders it harmless.

```

112 <latexrelease>\IncludeInRelease{0000/00/00}%
113 <latexrelease> {\@kernel@before@cenddocument}{kernel before hook}%
114 <latexrelease>
115 <latexrelease>\let\@kernel@before@cenddocument\c@empty
116 <latexrelease>
117 <latexrelease>\EndIncludeInRelease
118 <*2ekernel>

```

(End definition for \@kernel@before@cenddocument.)

```

\@testdef
119 \def\@testdef #1#2#3{%
120 \def\reserved@a{#3}\expandafter \ifx \csname #1#2\endcsname
121 \reserved@a \else \c@tempswatrue \fi}

```

(End definition for \@testdef.)

Reading data from auxiliary files (like .toc normally happens in vertical mode and it therefore doesn't matter if line endings are converted to spaces by TeX during that process.

However, especially the .toc file might be read in L-R mode (in cases the \tableofcontents attempts to put, say a list of sub-sections as a paragraph. In that case the newlines after a line like

```
\contentsline {subsubsection}{\numberline {1.1.1}A C-head}{2}
```

might result in spurious spaces (e.g., when that level is not included).

That could be fixed by reading in the file using `\endlinechar=-1` but that has the danger that it drops some valid endlines that should be converted to spaces (for example when the user edited the TOC and then used `\nofiles` to preserve it).

So the approach taken instead is this:

- `\addcontentsline` adds the command `\protected@file@percent` to the end of the second argument of `\@writefile` that is written to the `.aux`. As the name indicates this is a protected macro so it doesn't change if it is written out.
- When the `.aux` is read back in at the end of the run, `\@writefile` is executed and writes its second argument unmodified to the file with the extension given by its first argument. Or rather that was how it was in the past.
- Instead we change `\@writefile` slightly: basically it looks at the second argument and if the last token in there is `\protected@file@percent` then it is replaced by a percent character and that is then written out. If not (for example, if the data came from a user issued `\addtocontents`, or from some package that uses `\@writefile` for writing its own files) then the command behaves exactly as before.

`\protected@file@percent` Dummy cs to be replaced by a percent sign inside `\@writefile`. If it survives (when used incorrectly) it will expand to nothing in a typesetting context.

```
122 /2ekernel
123 {*2ekernel | latexrelease}
124 \latexrelease\IncludeInRelease{2018/12/01}%
125 \latexrelease%{\protected@file@percent}{Mask line endings}%
126 \protected\def\protected@file@percent{}
```

(End definition for `\protected@file@percent`.)

`\add@percent@to@temptokena` Helper function which is used to inspect a sequence of tokens (the second argument of `\@writefile` and if the last token is `\protected@file@percent` it will replace it by a harmless percent. The result is saved in `\@temptokena` for later use.

```
127 \catcode`^=9
128 \long\gdef\add@percent@to@temptokena#1{\protected@file@percent}\add@percent@to@temptokena
```

When we call this macro in `\@writefile` we stick in `\empty` at the beginning, so that in case the tokenlist consists of a single brace group the braces aren't stripped. The `\expandafter` then expands this extra token away again.

```
130 {\expandafter\ifx\expandafter X\detokenize{#2}X\expandafter\dont@add@percent@to@temptokena
131 \expandafter\do@add@percent@to@temptokena\fi{#1}}
132 \long\def\dont@add@percent@to@temptokena#1{\%
133 \@temptokena\expandafter{#1}}
```

`\latexrelease` will read this code in high-speed mode in certain situations. During that it will only look for `\if` tests but not actually execute the `\catcode` change above. As a result it will drop anything after the `%` character in the definition. Therefore the `\fi` needs to be on the next line and we need locally another comment character to avoid getting spaces into the definition—a weird problem :-)

```
134 \begingroup
```

```

135 \catcode`\%=12
136 \catcode`\^A=14
137 \long\gdef\do@add@percent@to@temptokena#1{\@temptokena\expandafter{\#1%`^A

```

Can't be on the same line as the % — see above.

```

138 }
139 \endgroup

```

(End definition for \add@percent@to@temptokena.)

\@writefile

```

140 \long\def\@writefile#1#2{%
141 \@ifundefined{tf@#1}\relax
142 {%

```

If we write to the file we first prepare #2 using \add@percent@to@temptokena and then write the token register out.

```

143 \add@percent@to@temptokena
144 \@empty#2\protected@file@percent
145 \add@percent@to@temptokena
146 \immediate\write\csname tf@#1\endcsname{\the\@temptokena}%
147 }%
148 }

149 </2ekernel | latexrelease>
150 <latexrelease>\EndIncludeInRelease
151 <latexrelease>\IncludeInRelease{0000/00/00}%
152 <latexrelease> {\protected@file@percent}{Mask line endings}%
153 <latexrelease>\let\protected@file@percent@\undefined
154 <latexrelease>\let\add@percent@to@temptokena@\undefined
155 <latexrelease>\let\do@add@percent@to@temptokena@\undefined
156 <latexrelease>\let\dont@add@percent@to@temptokena@\undefined
157 <latexrelease>\long\def\@writefile#1#2{%
158 \add@percent@to@temptokena
159 \relax
160 \immediate\write\csname tf@#1\endcsname{\the\@temptokena}%
161 }%
162 <latexrelease>}
163 <latexrelease>\EndIncludeInRelease
164 <*2ekernel>

```

(End definition for \@writefile.)

\stop

```

165 \def\stop{\clearpage\deadcycles\z@\let\par\@@par\@@end}

```

(End definition for \stop.)

*Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*

```

166 \everypar{\@nodocument} %% To get an error if text appears before the
167 \nullfont %% \begin{document}

```

\begin, \end, and \@checkend changed so \end{document} will catch an unmatched \begin. Changed 24 May 89 as suggested by Frank Mittelbach and Rainer Sch\"opf.

```

\begin{NAME} ==
BEGIN
 IF \NAME undefined THEN \reserved@a == BEGIN report error END
 ELSE \reserved@a ==
 (@currenvir :=L NAME) \NAME
 FI
 @ignore :=G F %% Added 30 Nov 88
 \begingroup
 \endpe := F
 \currenvir :=L NAME
 \NAME
END

\end{NAME} ==
BEGIN
 \endNAME
 \@checkend{NAME}
 \endgroup
 IF @endpe = T %% @endpe set True by \endparenv
 THEN \doendpe %% \doendpe redefines \par and \everypar
 %% to suppress paragraph indentation in
 %% immediately following text
 FI
 IF @ignore = T
 THEN @ignore :=G F
 \ignorespaces
 FI
END

\@checkend{NAME} ==
BEGIN
 IF \currenvir = NAME
 ELSE \@badend{NAME}
 FI
END

```

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

```

\begin
 168 </2ekernel>
 169 <*2ekernel | latexrelease>
 170 <| latexrelease>\IncludeInRelease{2020/10/01}%
 171 <| latexrelease> {\begin}{Use hook system}%
 172 \DeclareRobustCommand*\begin[1]{%
 173 \UseHook{env/#1/before}%
 174 \@ifundefined{#1}%

```

```

175 {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
176 {\def\reserved@a{\def\currenvir{#1}%
177 \edef\currenvline{\on@line}%
178 \@execute@begin@hook{#1}%
179 \csname #1\endcsname}%
180 \@ignorefalse
181 \begingroup\@endpefalse\reserved@a}

```

Before the `\document` code is executed we have to first undo the `\endgroup` as there should be none for this environment to avoid that changes on top-level unnecessarily go to TeX's savestack, and we have to initialize all hooks in the hook system. So we need to test for this environment name. But once it has been found all this testing is no longer needed and so we redefine `\@execute@begin@hook` to simply use the hook.

```

182 \def\@execute@begin@hook #1{%
183 \expandafter\ifx\csname #1\endcsname\document
184 \endgroup
185 \gdef\@execute@begin@hook##1{\UseHook{env/##1/begin}}%
186 \@expl@@@initialize@all@@
187 \fi

```

If this is an environment before `\begin{document}` we just run the hook so this can be outside the test.

```

188 \UseHook{env/#1/begin}%
189 }

```

The top level definition for `\end`. for an explanation see below (this is the same as the 2019 version where it was introduced, but for rollback we have to repeat it).

```

190 \edef\end
191 {\unexpanded{%
192 \romannumeral
193 \ifx\protect\@typeset@protect
194 \expandafter %1
195 \expandafter %2
196 \expandafter %1
197 \expandafter %3 expands the \csname inside \end<space>
198 \expandafter %1
199 \expandafter %2 expands \end<space>
200 \expandafter %1 expands the \else
201 \z@
202 \else
203 \expandafter\z@\expandafter\protect
204 \fi
205 }%
206 \expandafter\noexpand\csname end \endcsname
207 }

```

Version that adds hooks (so different from the 2019 version). It fixes tlb3722 but the change should perhaps be made in `tabularx` instead.

```

208 \Qnamedef{end }#1{%
209 \romannumeral
210 \IfHookEmptyTF{env/#1/end}%
211 {\expandafter\z@}%
212 {\z@\UseHook{env/#1/end}}%
213 \csname end#1\endcsname\@checkend{#1}%
214 \expandafter\endgroup\if@endpe\@doendpe\fi

```

```

215 \UseHook{env/#1/after}%
216 \if@ignore\@ignorefalse\ignorespaces\fi
217 }

```

Version without the fix for tlb3722 for the record:

```

218 \%@\namedef{end }#1{%
219 % \UseHook{env/#1/end}%
220 % \csname end#1\endcsname\@checkend{#1}%
221 % \expandafter\endgroup\if@endpe\@doendpe\fi
222 % \UseHook{env/#1/after}%
223 % \if@ignore\@ignorefalse\ignorespaces\fi}%
224 </2ekernel | latexrelease>
225 <latexrelease>\EndIncludeInRelease
226 <latexrelease>\IncludeInRelease[2019/10/01]%
227 <latexrelease> {\begin}{Making \begin/\end robust}%
228 <latexrelease>\DeclareRobustCommand\begin[1]{%
229 <latexrelease> \@ifundefined{#1}%
230 <latexrelease> {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
231 <latexrelease> {\def\reserved@a{\def\@currenvir{#1}}%
232 <latexrelease> \edef\@currenvline{\on@line}%
233 <latexrelease> \csname #1\endcsname}%
234 <latexrelease> \ignorespaces
235 <latexrelease> \begingroup\@endpefalse\reserved@a}

```

A version that doesn't start out with `\relax` when in typesetting mode would be the following, but since `\begin` issues a `\begingroup` it wouldn't help much with respect to allowing things like `\noalign` or `\multicolumn` inside.

```

236 \%edef\begin
237 % {\unexpanded{%
238 % \ifx\protect\@typeset@protect
239 % \expandafter\@gobble
240 % \fi
241 % \protect
242 % }%
243 % \expandafter\noexpand\csname begin \endcsname
244 % }
245 \%@\namedef{begin }#1{%
246 % \ifundefined{#1}%
247 % {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
248 % {\def\reserved@a{\def\@currenvir{#1}}%
249 % \edef\@currenvline{\on@line}%
250 % \csname #1\endcsname}%
251 % \ignorespaces
252 % \begingroup\@endpefalse\reserved@a}

```

`\end` While `\begin` was made robust simply by using `\DeclareRobustCommand` we need to be a bit more subtle with `\end` as there are packages out there that try to look into the top-level contents of `\end{foo}` (that is at the expansion of `\endfoo`) to see if it contains certain macros. This is done by hitting `\end{foo}` with three `\expandafters`, the first to get

```
\csname endfoo\endcsname \@checkend{foo}%
etc.
```

the second to expand the `\csname`, i.e., to get to

```
\endfoo \@checkend{foo}%
etc.
```

and the third to finally get to the top-level content of `\endfoo`, i.e.

```
<top-level content of \endfoo> \@checkend{foo}%
etc.
```

Therefore a robust replacement should produce the same results after three expansions (there first is obviously different).

Basically the definition of `\end` should either produce `\protect\end` (when not doing typesetting) or it should produce `\end` (without the `\protect`) when doing typesetting. Furthermore, it should (when in typesetting mode) show exactly the same result as `\end` (which is the original fragile definition of `\end`) when you expand either of them twice, i.e.,

```
\endfoo \@checkend{foo}%
etc.
```

That is achieved with the code below (which is worth studying carefully).

There is some trickery involved here: in particular we use `\romannumeral` to change a single expansion into three successive expansions in one go. That primitive expands until it has scanned a number (0 in this case, so it doesn't produce any output) and so it allows us to place arbitrary many `\expandafters` inside that are all going to be executed when `\romannumeral` is hit by a single `\expandafter`.

```
253 <latexrelease>\edef\end
254 <latexrelease> {\unexpanded{%
255 <latexrelease> \romannumeral
256 <latexrelease> \ifx\protect\@typeset\protect
257 <latexrelease> \expandafter %1
258 <latexrelease> \expandafter %2
259 <latexrelease> \expandafter %1
260 <latexrelease> \expandafter %3 expands the \csname inside \end<space>
261 <latexrelease> \expandafter %1
262 <latexrelease> \expandafter %2 expands \end<space>
263 <latexrelease> \expandafter %1 expands the \else
264 <latexrelease> \z@
265 <latexrelease> \else
266 <latexrelease> \expandafter\z@\expandafter\protect
267 <latexrelease> \fi
268 <latexrelease> }%
269 <latexrelease> \expandafter\noexpand\csname end \endcsname
270 <latexrelease> }
```

And here is the original definition of `\end` the way it was in L<sup>A</sup>T<sub>E</sub>X for several decades now hidden in `\end`.

```
271 <latexrelease>\@namedef{end }#1{%
272 <latexrelease> \csname end#1\endcsname\@checkend{#1}%
273 <latexrelease> \expandafter\endgroup\if@endpe\@doendpe\fi
274 <latexrelease> \if@ignore\@ignorefalse\ignorespaces\fi}
275 <latexrelease>\EndIncludeInRelease
```

An here the rollback in case that is ever needed.

```
276 <latexrelease>\IncludeInRelease{0000/00/00}%
277 <latexrelease> {\begin}{Making \begin/\end robust}%
278 <latexrelease>\def\begin#1{%
279 <latexrelease> \@ifundefined{#1}{%
```

```

280 <latexrelease> {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
281 <latexrelease> {\def\reserved@a{\def\@currenvir{#1}}%
282 <latexrelease> \edef\@currenvline{\on@line}%
283 <latexrelease> \csname #1\endcsname}%
284 <latexrelease> \ignorespaces
285 <latexrelease> \begingroup\endgroup\reserved@a
286 <latexrelease>\def\end#1{%
287 <latexrelease> \csname end#1\endcsname\@checkend{#1}%
288 <latexrelease> \expandafter\endgroup\if\endpe\@doendpe\fi
289 <latexrelease> \if\ignore\ignorespaces\fi}
290 <latexrelease>

```

Also undo the internal commands as some packages unfortunately test for their existence instead of using `\IfFormatAtLeastTF`.

```

291 <latexrelease>\expandafter\let\csname begin \endcsname\@undefined
292 <latexrelease>\expandafter\let\csname end \endcsname\@undefined
293 <latexrelease>
294 <latexrelease>\EndIncludeInRelease
295 {*2ekernel}

```

(*End definition for `\begin` and `\end`.*)

```

\@checkend
296 \def\@checkend#1{\def\reserved@a{#1}\ifx
297 \reserved@a\@currenvir \else\@badend{#1}\fi}

```

(*End definition for `\@checkend`.*)

**\@currenvline** We do need a default value for `\@currenvline` on top-level since the document environment cancels the brace group. This means that a mismatch with `\begin{document}` will not produce a line number. Thus the outer default must be `\@empty` or we will end up with two spaces.

```
298 \let\@currenvline\@empty
```

(*End definition for `\@currenvline`.*)

**\AtBeginEnvironment** We provide 4 high-level hook interfaces directly, the others only when etoolbox is loaded

```

\AtEndEnvironment
\BeforeBeginEnvironment
\AfterEndEnvironment
299 {*}2ekernel}
300 {*2ekernel | latexrelease}
301 <latexrelease>\IncludeInRelease{2020/10/01}%
302 <latexrelease> {\AtBeginEnvironment}{Hooks for environments}%
303 \newcommand\AtBeginEnvironment[2] [...] {\AddToHook{env/#2/begin}[]{#1}}
304 \newcommand\AtEndEnvironment[2] [...] {\AddToHook{env/#2/end}[]{#1}}
305 \newcommand\BeforeBeginEnvironment[2] [...] {\AddToHook{env/#2/before}[]{#1}}
306 \newcommand\AfterEndEnvironment[2] [...] {\AddToHook{env/#2/after}[]{#1}}
307 {*}2ekernel | latexrelease}
308 <latexrelease>\EndIncludeInRelease
309 <latexrelease>\IncludeInRelease{0000/00/00}%
310 <latexrelease> {\AtBeginEnvironment}{Hooks for environments}%
311 <latexrelease>
312 <latexrelease>\let\AtBeginEnvironment\@undefined
313 <latexrelease>\let\AtEndEnvironment\@undefined
314 <latexrelease>\let\BeforeBeginEnvironment\@undefined

```

```

315 ⟨latexrelease⟩\let\AfterEndEnvironment\@undefined
316 ⟨latexrelease⟩
317 ⟨latexrelease⟩\EndIncludeInRelease
318 ⟨*2ekernel⟩

```

(*End definition for \AtBeginEnvironment and others. These functions are documented on page 200.*)

## 1.2 Center, Flushright, Flushleft

```

319 \message{center,}

```

*Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*

```

\center, \flushright and \flushleft set
\rightskip = 0pt or \@flushglue (as appropriate)
\leftskip = 0pt or \@flushglue (as appropriate)
\parindent = 0pt
\parfillskip = 0pt. (except \flushleft)
\\ == \par \vskip -\parskip
\\[LENGTH] == \\ \vskip LENGTH
* == \par \penalty 10000 \vskip -\parskip
*[LEN] == * \vskip LENGTH

```

They invoke the trivlist environment to handle vertical spacing before and after them.

\centering, \raggedright and \raggedleft are the declaration analogs of the above.

\raggedright has a more universal effect, however. It sets \rightskip := flushglue. Every environment, like the list environments, that set \rightskip to its 'normal' value set it to \rightskip

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

```
\@centercr
```

```

320 ⟨/2ekernel⟩
321 ⟨*2ekernel | latexrelease⟩
322 ⟨latexrelease⟩\IncludeInRelease{2020/02/02}%
323 ⟨latexrelease⟩ {\@centercr}{\Make robust}%
324 \protected\def\@centercr{\ifhmode \unskip\else \nolnerr\fi
325 \par\@ifstar{\nobreak\@xcentercr}\@xcentercr}
326 ⟨/2ekernel | latexrelease⟩

327 ⟨latexrelease⟩\EndIncludeInRelease
328 ⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
329 ⟨latexrelease⟩ {\@centercr}{\Make robust}%
330 ⟨latexrelease⟩
331 ⟨latexrelease⟩\def\@centercr{\ifhmode \unskip\else \nolnerr\fi
332 ⟨latexrelease⟩ \par\@ifstar{\nobreak\@xcentercr}\@xcentercr}
333 ⟨latexrelease⟩
334 ⟨latexrelease⟩\EndIncludeInRelease
335 ⟨*2ekernel⟩

```

(End definition for \@centercr.)

\@xcentercr

```
336 \def\@xcentercr{\addvspace{-\parskip}\@ifnextchar
337 [\@icentercr\ignorespaces}
```

(End definition for \@xcentercr.)

\@icentercr

```
338 </2ekernel>
339 <*2ekernel | latexrelease>
340 <latexrelease>\IncludeInRelease{2020/10/01}%
341 <latexrelease> {\@icentercr}{centering, etc support calc}%
342 \def\@icentercr[#1]{\vspace@calcify{#1}\ignorespaces}
343 </2ekernel | latexrelease>
344 <latexrelease>\EndIncludeInRelease
345 <latexrelease>\IncludeInRelease{0000/00/00}%
346 <latexrelease> {\@icentercr}{centering, etc support calc}%
347 <latexrelease>
348 <latexrelease>\def\@icentercr[#1]{\vskip #1\ignorespaces}
349 <latexrelease>\EndIncludeInRelease
350 <*2ekernel>
```

(End definition for \@icentercr.)

**center** (*env.*) We use \relax to prevent \item scanning too far.

```
351 \def\centerf{\trivlist \centering\item\relax}
352 \def\endcenter{\endtrivlist}
353 </2ekernel>
354 <*2ekernel | latexrelease>
355 <latexrelease>\IncludeInRelease{2020/10/01}%
356 <latexrelease> {\centering}{Set finalhyphendemerits}%
```

\centering

```
357 \DeclareRobustCommand\centering{%
358 \let\\@\centercr
359 \rightskip\flushglue\leftskip\flushglue
360 \finalhyphendemerits=z@
361 \parindent z@\parfillskip z@skip}
```

(End definition for \centering.)

\raggedright

```
362 \DeclareRobustCommand\raggedright{%
363 \let\\@\centercr\rightskip\flushglue \rightskip\rightskip
364 \finalhyphendemerits=z@
365 \leftskip z@skip
366 \parindent z@}
```

(End definition for \raggedright.)

```

\raggedleft
367 \DeclareRobustCommand\raggedleft{%
368 \let\\@centercr
369 \rightskip\z@skip\leftskip\@flushglue
370 \finalhyphendemerits=\z@
371 \parindent\z@\parfillskip\z@skip}

(End definition for \raggedleft.)

372 </2ekernel | latexrelease>
373 <latexrelease>\EndIncludeInRelease
374 <latexrelease>\IncludeInRelease{2019/10/01}%
375 <latexrelease> {\centering}{Make commands robust}%
376 <latexrelease>
377 <latexrelease>\DeclareRobustCommand\centering{%
378 \let\\@centercr
379 \rightskip\@flushglue\leftskip\@flushglue
380 \parindent\z@\parfillskip\z@skip}
381 <latexrelease>\DeclareRobustCommand\raggedright{%
382 \let\\@centercr\rightskip\@flushglue \rightskip\@rightskip
383 \leftskip\z@skip
384 \parindent\z@}
385 <latexrelease>\DeclareRobustCommand\raggedleft{%
386 \let\\@centercr
387 \rightskip\z@skip\leftskip\@flushglue
388 \parindent\z@\parfillskip\z@skip}
389 <latexrelease>\EndIncludeInRelease
390 <latexrelease>
391 <latexrelease>\IncludeInRelease{0000/00/00}%
392 <latexrelease> {\centering}{Make commands robust}%
393 <latexrelease>
394 <latexrelease>\kernel@make@fragile\centering
395 <latexrelease>\kernel@make@fragile\raggedright
396 <latexrelease>\kernel@make@fragile\raggedleft
397 <latexrelease>
398 <latexrelease>\EndIncludeInRelease
399 <*2ekernel>

@rightskip
400 \newskip\@rightskip \rightskip \z@skip

(End definition for \@rightskip.)

flushleft (env.) We use \relax to prevent \item scanning too far.
401 \def\flushleft{\trivlist \raggedright\item\relax}
402 \def\endflushleft{\endtrivlist}

flushright (env.) We use \relax to prevent \item scanning too far.
403 \def\flushright{\trivlist \raggedleft\item\relax}
404 \def\endflushright{\endtrivlist}

```

### 1.3 Verbatim

405 \message{verbatim,}

The verbatim environment uses the fixed-width `\ttfamily` font, turns blanks into spaces, starts a new line for each carriage return (or sequence of consecutive carriage returns), and interprets *every* character literally. I.e., all special characters `\`, `{`, `$`, etc. are `\catcode`'d to 'other'.

The command `\verb` produces in-line verbatim text, where the argument is delimited by any pair of characters. E.g., `\verb #...#` takes '...' as its argument, and sets it verbatim in `\ttfamily` font.

The `*`-variants of these commands are the same, except that spaces print as the TeXbook's space character instead of as blank spaces.

`\@vobeyspaces`

406 {\catcode`\\ =\active%  
407 \gdef\@vobeyspaces{\catcode`\\ \active\let \xobeysp{}}

(End definition for `\@vobeyspaces`.)

`\@xobeysp`

(End definition for `\@xobeysp`.)

`\@xverbatim`

`\@sxverbatim` 408 \begingroup \catcode '|=0 \catcode '['= 1  
409 \catcode']=2 \catcode '\{|=12 \catcode '\}=12  
410 \catcode'\|=12 \gdef\@xverbatim#1\end{verbatim}#[#1|end[verbatim]]  
411 \gdef\@sxverbatim#1\end{verbatim\*}#[#1|end[verbatim\*]]  
412 |endgroup

(End definition for `\@xverbatim` and `\@sxverbatim`.)

`\@verbatim` Real start of verbatim environment We use `\relax` to prevent `\item` scanning too far.

413 </2ekernel>  
414 <\*2ekernel | latexrelease>  
415 <latexrelease>\IncludeInRelease{2017-04-15}{\@verbatim}%  
416 <latexrelease> {Disable hyphenation in verbatim}%  
417 \def\@verbatim{\trivlist \item\relax  
418 \if@minipage\else\vskip\parskip\fi  
419 \leftskip\@totalleftmargin\rightskip\z@skip  
420 \parindent\z@\parfillskip\@flushglue\parskip\z@skip

Added `\@par` to clear possible `\parshape` definition from a surrounding list (the verbatim guru says). Switch language when in vertical mode.

421 \@par

Set `\language` here to suppress hyphenation. Done this way rather than setting `\hyphenchar` as that is a global setting.

422 \language\l@nohyphenation  
423 \tempswafalse  
424 \def\par{  
425 \if@tempswa

A `\leavevmode` added: needed if, for example, a blank verbatim line is the first thing in a list item (wow!).

```

426 \leavevmode \null \@@par\penalty\interlinepenalty
427 \else
428 \ctempswattrue
429 \ifhmode\@@par\penalty\interlinepenalty\fi
430 \fi}%

```

To allow customization we hide the font used in a separate macro.

```

431 \let\do\@makeother \dospecials
432 \obeylines \verbatim@font \noligs

```

To avoid a breakpoint after the labels box, we remove the penalty put there by the list macros: another use of `\unpenalty!`

```

433 \everypar \expandafter{\the\everypar \unpenalty}%
434 }
435 </2ekernel | latexrelease>
436 <latexrelease>\EndIncludeInRelease
437 <latexrelease>\IncludeInRelease{0000-00-00}{\overbatim}%
438 <latexrelease> {Disable hyphenation in verbatim}%
439 <latexrelease>\def\overbatim{\trivlist \item\relax
440 <latexrelease> \if@minipage\else\vskip\parskip\fi
441 <latexrelease> \leftskip\@totalleftmargin\rightskip\z@skip
442 <latexrelease> \parindent\z@\parfillskip\@flushglue\parskip\z@skip
443 <latexrelease> \@@par
444 <latexrelease> \ctempswafalse
445 <latexrelease> \def\par{%
446 <latexrelease> \if@tempswa
447 <latexrelease> \leavevmode \null \@@par\penalty\interlinepenalty
448 <latexrelease> \else
449 <latexrelease> \ctempswattrue
450 <latexrelease> \ifhmode\@@par\penalty\interlinepenalty\fi
451 <latexrelease> \fi}%
452 <latexrelease> \let\do\@makeother \dospecials
453 <latexrelease> \obeylines \verbatim@font \noligs
454 <latexrelease> \hyphenchar\font\m@ne
455 <latexrelease> \everypar \expandafter{\the\everypar \unpenalty}%
456 <latexrelease> }
457 <latexrelease>\EndIncludeInRelease
458 {*2ekernel}

```

*(End definition for `\overbatim`.)*

**`\verb+verbatim+`** (RmS 93/09/19) Protected against ‘missing item’ error message triggered by empty `\endverbatim` environment.

```

459 \def\verbatim{\overbatim \frenchspacing\obeyspaces \xoverbatim}
460 \def\endverbatim{\if@newlist \leavevmode\fi\endtrivlist}

```

*(End definition for `\verbatim` and `\endverbatim`.)*

**`\verb+verbatim@font+`** Macro to select the font used for verbatim typesetting. It also does other work if necessary for the font used.

```

461 \def\verbatim@font{\normalfont\ttfamily}

```

```

(End definition for \verb@font.))

462 </2ekernel>
463 <*2ekernel | latexrelease>
464 <latexrelease>\IncludeInRelease{2018/12/01}%
465 <latexrelease> {\verbvisible}{Setup visible space for \verb}%

```

**\asciispace** The character in slot 32, in typewriter fonts (historically) a visible space but in other fonts a real space or something else

```

466 \DeclareRobustCommand\asciispace{\char 32 }

(End definition for \asciispace.)
```

**\verbvisible** This defines how to get a visible space in \verb\* and friends. In classic TeX this is just the slot 32, but in TU encoded fonts we switch fonts and take the character from cmtt.

```

467 \ifx\Umathcode\@undefined
468 \let\verbvisible\asciispace % PdfTeX version
469 \else
470 \DeclareRobustCommand\verbvisible
471 {\leavevmode{\usefont{OT1}{cmtt}{m}{n}\asciispace}} % xetex/luatex version
472 \fi

(End definition for \verbvisible.)
```

**\@setupverbvisible** In pdfTeX a catcode 12 space will produce the character in slot 32 which is assumed to be a visible space character (in a typewriter font in OT1 or T1 encoding). In XeTeX or LuaTeX a font in TU encoding is normally used and that has a real space in this slot. So what we do in this case is this: we check the definition of \verbvisible and if it is \asciispace we assume that the char32 can be used (e.g., in pdfTeX). We then redefine \xobeysp so that after running \@vobeyspaces we get characters from slot 32 for each active space.

```

473 \def\@setupverbvisible{%
474 \ifx\verbvisible\asciispace
475 \let\xobeysp\asciispace
476 \else
```

Otherwise we measure the width of a character in the mon-spaced current font and place a \verbvisible into a box of the right width which we are then using as the character for a space. By default this will be the space character from OT1 cmtt but by changing \verbvisible one could use, for example, the \textvisible of the current typewriter font.

```

477 \setbox\z@\hbox{x}%
478 \setbox\@verbvisiblebox\hbox to\wd\z@{\hss\verbvisible\hss}%
479 \def\xobeysp{\leavevmode\copy\@verbvisiblebox}%
480 \fi
481 }
```

(End definition for \@setupverbvisible.)

**\@verbvisiblebox** The box to hold the visible space character if it isn't in slot 32 in the current typewriter font.

```

482 \newbox\@verbvisiblebox
```

(End definition for \@verbvisiblebox.)

**verbatim\*** (*env.*) For **verbatim\*** we also set up the correct visible space character definition and then run **\@vobeyspaces**. As this code is not called as part of the normal verbatim environment (the method is done the other way around this time) we don't have to check if space is already active—it shouldn't be.

```

483 \@namedef{verbatim*}{\@verbatim
484 \@setupverbvisiblespace
485 \frenchspacing\@vobeyspaces\@sxverbatim}
486 \expandafter\let\csname endverbatim*\endcsname =\endverbatim

487 </2ekernel | latexrelease>
488 \<latexrelease>\EndIncludeInRelease
489 \<latexrelease>\IncludeInRelease{0000/00/00}%
490 \<latexrelease> {\verbvisible}{Setup visible space for \verb}%
491 \<latexrelease>
492 \<latexrelease>\@namedef{verbatim*}{\@verbatim\@sxverbatim}
493 \<latexrelease>
494 \<latexrelease>\let\asciispace \@undefined
495 \<latexrelease>\let\verbvisiblespace \@undefined
496 \<latexrelease>\let\@setupverbvisiblespace\@undefined
497 \<latexrelease>\let\@verbvisiblespacebox \@undefined
498 \<latexrelease>\EndIncludeInRelease
499 {*2ekernel}

```

**\@sverb** Definitions of **\@sverb** and **\@verb** changed so **\verb+ foo+** does not lose leading blanks  
**\@@sverb** when it comes at the beginning of a line. Change made 24 May 89. Suggested by Frank  
Mittelbach and Rainer Schöpf.

```

500 </2ekernel>
501 {*2ekernel | latexrelease}
502 \<latexrelease>\IncludeInRelease{2020/10/01}%
503 \<latexrelease> {\@sverb}{Drop spaces before \verb delimiter}%

```

If the users types **\verb !~! foo** then surprisingly we would get the space as the delimiter and thus “**!~!foo**” in the output. To avoid this scenario we check if #1 has the character code of a space, if so we recurse otherwise we call **\@@sverb** (which is the original definition of **\@sverb**).

```

504 \def\@sverb#1{\if\noexpand#1 \expandafter\@sverb\else\@@sverb{#1}\fi}

505 \def\@@sverb#1{%
506 \catcode`#1\active
507 \lccode`\~`#1%
508 \gdef\verb@balance@group{\verb@egroup
509 \@latex@error{\noexpand\verb illegal in argument}\@ehc}%
510 \aftergroup\verb@balance@group
511 \lowercase{\let\~\verb@egroup}%

```

If **\@sverb** is called from **\@verb** then space is already active and supposed to produce a real space. In this case we do nothing. Otherwise we run **\@setupverbvisiblespace** to setup the right visible space char and afterwards **\@vobeyspaces** to make it the definition for the active space character.

```

512 \ifnum\catcode`\ =\active
513 \else \@setupverbvisiblespace \@vobeyspaces \fi
514 }

```

```

515 </2ekernel | latexrelease>
516 <latexrelease>\EndIncludeInRelease
517 <latexrelease>\IncludeInRelease{2018/12/01}%
518 <latexrelease> {\@sverb}{Setup visible space for \verb}%
519 <latexrelease>
520 <latexrelease>\def\@sverb#1{%
521 <latexrelease> \catcode`#1\active
522 <latexrelease> \lccode`\~`#1%
523 <latexrelease> \gdef\verb@balance@group{\verb@egroup
524 <latexrelease> \o@latex@error{\noexpand\verb illegal in command argument}\o@ehc}%
525 <latexrelease> \aftergroup\verb@balance@group
526 <latexrelease> \lowercase{\let~\verb@egroup}%
527 <latexrelease> \ifnum\catcode`\ =\active
528 <latexrelease> \else \o@setupverbvisibleSpace \o@vobeyspaces \fi
529 <latexrelease>}
530 <latexrelease>\let\o@sverb\o@undefined
531 <latexrelease>\EndIncludeInRelease
532 <latexrelease>
533 <latexrelease>\IncludeInRelease{0000/00/00}%
534 <latexrelease> {\@sverb}{Setup visible space for \verb}%
535 <latexrelease>\def\@sverb#1{%
536 <latexrelease> \catcode`#1\active
537 <latexrelease> \lccode`\~`#1%
538 <latexrelease> \gdef\verb@balance@group{\verb@egroup
539 <latexrelease> \o@latex@error{\noexpand\verb illegal in command argument}\o@ehc}%
540 <latexrelease> \aftergroup\verb@balance@group
541 <latexrelease> \lowercase{\let~\verb@egroup}%
542 <latexrelease>
543 <latexrelease>\EndIncludeInRelease
544 {*2ekernel}

```

(End definition for \o@sverb and \o@osverb.)

\@makeother

```
545 \def\@makeother#1{\catcode`#112\relax}
```

(End definition for \@makeother.)

\verb@balance@group

```
546 \let\verb@balance@group\empty
```

(End definition for \verb@balance@group.)

\verb@egroup

```
547 \def\verb@egroup{\global\let\verb@balance@group\empty\egroup}
```

(End definition for \verb@egroup.)

\verb@eol@error

```

548 \begingroup
549 \obeylines%
550 \gdef\verb@eol@error{\obeylines%
551 \def^~M{\verb@egroup\o@latex@error{%
552 \noexpand\verb ended by end of line}\o@ehc}}%
553 \endgroup

```

(End definition for \verb@eol@error.)

\verb Typesetting a small piece verbatim.

```
554 </2ekernel>
555 {*2ekernel | latexrelease}
556 <latexrelease>\IncludeInRelease{2017-04-15}{\verb}%
557 <latexrelease> {Disable hyphenation in verb}%
558 \def\verb{\relax\ifmmode\hbox\else\leavevmode\null\fi
559 \bgroup
560 \verb@eol@error \let\do\@makeother \dospecials
561 \verbatim@font\@noligs
```

Set \language here to suppress hyphenation. Done this way rather than setting \hyphenchar as that is a global setting.

```
562 \language\l@nohyphenation
563 \@ifstar\@sverb\@verb}
564 </2ekernel | latexrelease>
565 <latexrelease>\EndIncludeInRelease
566 <latexrelease>\IncludeInRelease{0000-00-00}{\verb}%
567 <latexrelease> {Disable hyphenation in verb}%
568 <latexrelease>\def\verb{\relax\ifmmode\hbox\else\leavevmode\null\fi
569 <latexrelease> \bgroup
570 <latexrelease> \verb@eol@error \let\do\@makeother \dospecials
571 <latexrelease> \verbatim@font\@noligs
572 <latexrelease> \@ifstar\@sverb\@verb}
573 <latexrelease>\EndIncludeInRelease
574 {*2ekernel>
```

(End definition for \verb.)

\@verb

```
575 \def\@verb{\@vobeyspaces \frenchspacing \@sverb}
```

(End definition for \@verb.)

\verbatim@nolig@list

```
576 \def\verbatim@nolig@list{\do`\'\do`<\do`>\do`,\do`'\do`-`}
```

(End definition for \verbatim@nolig@list.)

\do@noligs

```
577 \def\do@noligs#1{%
578 \catcode`#1\active
579 \begingroup
580 \lccode`\~`#1\relax
581 \lowercase{\endgroup\def~{\leavevmode\kern\z@\char`#1}}}
```

(End definition for \do@noligs.)

\@noligs To stay compatible with packages that use \@noligs we keep it.

```
582 \def\@noligs{\let\do\do@noligs \verbatim@nolig@list}
```

(End definition for \@noligs.)

```
583 </2ekernel>
```

# File I

## ltmath.dtx

### 1 Math setup

This file contains a lot of the original plain T<sub>E</sub>X code, as well as the L<sup>A</sup>T<sub>E</sub>X environments for math. It still needs sorting out.

```
1 <*2ekernel>
2 \message{math definitions,}
```

#### 1.1 Math commands based on plain T<sub>E</sub>X

##### 1.1.1 The log-like functions

\log The standard operators:

```
3 \DeclareRobustCommand\log{\mathop{\operator@font log}\nolimits}
4 \DeclareRobustCommand\lg{\mathop{\operator@font lg}\nolimits}
5 \DeclareRobustCommand\ln{\mathop{\operator@font ln}\nolimits}
6 \DeclareRobustCommand\lim{\mathop{\operator@font lim}\nolimits}
7 \DeclareRobustCommand\limsup{\mathop{\operator@font lim\,,sup}\nolimits}
8 \DeclareRobustCommand\liminf{\mathop{\operator@font lim\,,inf}\nolimits}
9 \DeclareRobustCommand\sin{\mathop{\operator@font sin}\nolimits}
10 \DeclareRobustCommand\arcsin{\mathop{\operator@font arcsin}\nolimits}
11 \DeclareRobustCommand\sinh{\mathop{\operator@font sinh}\nolimits}
12 \DeclareRobustCommand\cos{\mathop{\operator@font cos}\nolimits}
13 \DeclareRobustCommand\arccos{\mathop{\operator@font arccos}\nolimits}
14 \DeclareRobustCommand\cosh{\mathop{\operator@font cosh}\nolimits}
15 \DeclareRobustCommand\tan{\mathop{\operator@font tan}\nolimits}
16 \DeclareRobustCommand\arctan{\mathop{\operator@font arctan}\nolimits}
17 \DeclareRobustCommand\tanh{\mathop{\operator@font tanh}\nolimits}
18 \DeclareRobustCommand\cot{\mathop{\operator@font cot}\nolimits}
19 \DeclareRobustCommand\coth{\mathop{\operator@font coth}\nolimits}
20 \DeclareRobustCommand\sec{\mathop{\operator@font sec}\nolimits}
21 \DeclareRobustCommand\csc{\mathop{\operator@font csc}\nolimits}
22 \DeclareRobustCommand\max{\mathop{\operator@font max}\nolimits}
23 \DeclareRobustCommand\min{\mathop{\operator@font min}\nolimits}
24 \DeclareRobustCommand\sup{\mathop{\operator@font sup}\nolimits}
25 \DeclareRobustCommand\inf{\mathop{\operator@font inf}\nolimits}
26 \DeclareRobustCommand\arg{\mathop{\operator@font arg}\nolimits}
27 \DeclareRobustCommand\ker{\mathop{\operator@font ker}\nolimits}
28 \DeclareRobustCommand\dim{\mathop{\operator@font dim}\nolimits}
29 \DeclareRobustCommand\hom{\mathop{\operator@font hom}\nolimits}
30 \DeclareRobustCommand\det{\mathop{\operator@font det}\nolimits}
31 \DeclareRobustCommand\exp{\mathop{\operator@font exp}\nolimits}
32 \DeclareRobustCommand\Pr{\mathop{\operator@font Pr}\nolimits}
33 \DeclareRobustCommand\gcd{\mathop{\operator@font gcd}\nolimits}
34 \DeclareRobustCommand\deg{\mathop{\operator@font deg}\nolimits}
```

(End definition for \log.)

\bmod And some operators have to be done by hand:

```

35 \DeclareRobustCommand\bmod{%
36 \nonscript\mskip-\medmuskip\mkern5mu%
37 \mathbin{\operator@font mod}\penalty900\mkern5mu%
38 \nonscript\mskip-\medmuskip}

```

(End definition for `\bmod`.)

`\pmod`

```

39 \DeclareRobustCommand\pmod[1]{%
40 \allowbreak\mkern18mu(\operator@font mod}\,,\,#1)}

```

(End definition for `\pmod`.)

### 1.1.2 Biggggg

`\big` Variants on `\big` and friends for use with delimiters:

```

41 \DeclareRobustCommand\bigl{\mathopen\big}
42 \DeclareRobustCommand\bigm{\mathrel\big}
43 \DeclareRobustCommand\bigr{\mathclose\big}
44 \DeclareRobustCommand\Bigl{\mathopen\Big}
45 \DeclareRobustCommand\Bigm{\mathrel\Big}
46 \DeclareRobustCommand\Bigr{\mathclose\Big}
47 \DeclareRobustCommand\biggl{\mathopen\bigg}
48 \DeclareRobustCommand\biggm{\mathrel\bigg}
49 \DeclareRobustCommand\biggr{\mathclose\bigg}
50 \DeclareRobustCommand\Biggl{\mathopen\Bigg}
51 \DeclareRobustCommand\Biggm{\mathrel\Bigg}
52 \DeclareRobustCommand\Biggr{\mathclose\Bigg}

```

(End definition for `\big`.)

### 1.1.3 The UNSORTED Rest

The other math commands are lifted from plain TeX.

`\jot`

```

53 \newdimen\jot
54 \jot=3pt

```

(End definition for `\jot`.)

`\interdisplaylinepenalty`

```

55 \newcount\interdisplaylinepenalty
56 \interdisplaylinepenalty=100

```

(End definition for `\interdisplaylinepenalty`.)

`\choose`

```

57 \def\choose{\atopwithdelims()}

```

(End definition for `\choose`.)

`\brack`

```

58 \def\brack{\atopwithdelims[]}

```

(End definition for `\brack`.)

```

\brace
59 \def\brace{\atopwithdelims\{\}}
(End definition for \brace.)

\mathpalette
60 \def\mathpalette#1#2{%
61 \mathchoice
62 {#1\displaystyle{#2}}%
63 {#1\textstyle{#2}}%
64 {#1\scriptstyle{#2}}%
65 {#1\scriptscriptstyle{#2}}}
(End definition for \mathpalette.)

\root
\rootbox
66 \newbox\rootbox
\root@t
67 \def\root#1{\of{%
68 \setbox\rootbox\hbox{$\m@th\scriptscriptstyle{#1}$}%
69 \mathpalette\root@t}}
\def\root@t#1#2{%
70 \setbox\z@\hbox{$\m@th\sqrt{#2}$}%
71 \dimen@\ht\z@\advance\dimen@-\dp\z@
72 \mkern5mu\raise.6\dimen@\copy\rootbox
73 \mkern-10mu\box\z@}
(End definition for \root, \rootbox, and \root@t.)

\phantom
\hphantom
75 \newif\ifv@
\phantom
76 \newif\ifh@
77 </2ekernel>
78 <*2ekernel | latexrelease>
79 <latexrelease>\IncludeInRelease{2019/10/01}%
80 <latexrelease> {\vphantom}{Make commands robust}%
81 \DeclareRobustCommand\vphantom{\v@true\h@false\ph@nt}
82 \DeclareRobustCommand\hphantom{\v@false\h@true\ph@nt}
83 \DeclareRobustCommand

84 \DeclareRobustCommand\mathstrut{\vphantom{}}

\mathstrut
85 </2ekernel | latexrelease>
86 <latexrelease>\EndIncludeInRelease
87 <latexrelease>\IncludeInRelease{0000/00/00}%
88 <latexrelease> {\vphantom}{Make commands robust}%
89 <latexrelease>
90 <latexrelease>\kernel@make@fragile\vphantom
91 <latexrelease>\kernel@make@fragile\hphantom
92 <latexrelease>\kernel@make@fragile\phantom
93 <latexrelease>\kernel@make@fragile\mathstrut
94 <latexrelease>
95 <latexrelease>\EndIncludeInRelease
96 <*2ekernel>

```

```

97 \def\ph@nt{%
98 \ifmmode
99 \expandafter\mathpalette\expandafter\mathph@nt
100 \else
101 \expandafter\makeph@nt
102 \fi}
103 \def\makeph@nt#1{%
104 \setbox\z@\hbox{\color@begingroup#1\color@endgroup}\finph@nt}
105 \def\mathph@nt#1#2{%
106 \setbox\z@\hbox{$\m@th#1{#2}$}\finph@nt}
107 </2ekernel>
108 <*2ekernel | latexrelease>
109 <latexrelease>\IncludeInRelease{2018/12/01}%
110 <latexrelease> {\finph@nt}{Start LR-mode}%
111 \def\finph@nt{%
112 \setbox\tw@\null
113 \ifv@ \ht\tw@\ht\z@ \dp\tw@\dp\z@\fi
114 \ifh@ \wd\tw@\wd\z@\fi
115 \leavevemode@ifvmode\box\tw@}
116 </2ekernel | latexrelease>
117 <latexrelease>\EndIncludeInRelease
118 <latexrelease>\IncludeInRelease{0000/00/00}%
119 <latexrelease> {\finph@nt}{Start LR-mode}%
120 <latexrelease>\def\finph@nt{%
121 <latexrelease> \setbox\tw@\null
122 <latexrelease> \ifv@ \ht\tw@\ht\z@ \dp\tw@\dp\z@\fi
123 <latexrelease> \ifh@ \wd\tw@\wd\z@\fi \box\tw@}
124 <latexrelease>\EndIncludeInRelease
125 <*2ekernel>

```

(End definition for *\phantom* and others.)

```

\smash
126 \DeclareRobustCommand\smash{%
127 \relax % \relax, in case this comes first in \halign
128 \ifmmode
129 \expandafter\mathpalette\expandafter\mathsm@sh
130 \else
131 \expandafter\makesm@sh
132 \fi}
133 \def\makesm@sh#1{%
134 \setbox\z@\hbox{\color@begingroup#1\color@endgroup}\finsm@sh}
135 \def\mathsm@sh#1#2{%
136 \setbox\z@\hbox{$\m@th#1{#2}$}\finsm@sh}
137 </2ekernel>
138 <*2ekernel | latexrelease>
139 <latexrelease>\IncludeInRelease{2018/12/01}%
140 <latexrelease> {\finsm@sh}{Start LR-mode}%
141 \def\finsm@sh{\ht\z@\z@ \dp\z@\z@ \leavevemode@ifvmode\box\z@}
142 </2ekernel | latexrelease>
143 <latexrelease>\EndIncludeInRelease

```

```

144 \langle latexrelease \rangle \IncludeInRelease{0000/00/00}%
145 \langle latexrelease \rangle \finsm@sh{\StartLRMode}%
146 \langle latexrelease \rangle \def\finsm@sh{\ht\z@\z@ \dp\z@\z@ \box\z@}
147 \langle latexrelease \rangle \EndIncludeInRelease
148 \langle *2ekernel \rangle

(End definition for \smash.)
```

\buildrel

```

149 \def\buildrel#1\over#2{\mathrel{\mathop{\kern\z@#2}\limits^{#1}}}
```

(End definition for \buildrel.)

```

150 \langle /2ekernel \rangle
151 \langle *2ekernel | latexrelease \rangle
152 \langle latexrelease \rangle \IncludeInRelease{2019/10/01}%
153 \langle latexrelease \rangle \cases{Make commands robust}%
```

\cases

```

154 \DeclareRobustCommand*\cases[1]{\left\{\,\vcenter{\normalbaselines\m@th
155 \ialign{$##\hfil&\quad##\hfil\crcr#1\crcr}}\right.}
```

(End definition for \cases.)

\matrix

```

156 \DeclareRobustCommand*\matrix[1]{\null\,,\vcenter{\normalbaselines\m@th
157 \ialign{\hfil###\hfil&\quad\hfil###\hfil\crcr
158 \mathstrut\crcr\noalign{\kern-\baselineskip}
159 #1\crcr\mathstrut\crcr\noalign{\kern-\baselineskip}}}\,,}
```

(End definition for \matrix.)

\pmatrix

```

160 \DeclareRobustCommand*\pmatrix[1]{\left(\matrix{#1}\right)}
```

(End definition for \pmatrix.)

```

161 \langle /2ekernel | latexrelease \rangle
162 \langle latexrelease \rangle \EndIncludeInRelease
163 \langle latexrelease \rangle \IncludeInRelease{0000/00/00}%
164 \langle latexrelease \rangle \cases{Make commands robust}%
165 \langle latexrelease \rangle
166 \langle latexrelease \rangle \kernel@make@fragile\cases
167 \langle latexrelease \rangle \kernel@make@fragile\matrix
168 \langle latexrelease \rangle \kernel@make@fragile\pmatrix
169 \langle latexrelease \rangle
170 \langle latexrelease \rangle \EndIncludeInRelease
171 \langle *2ekernel \rangle
```

\bordermatrix

```

172 \def\bordermatrix#1{\begingroup \m@th
173 \tempdima 8.75\p@
174 \setbox\z@\vbox{%
175 \def\cr{\crcr\noalign{\kern2\p@\global\let\cr\endline}}%
176 \ialign{$##\hfil\kern2\p@\kern\tempdima\&\thinspace\hfil$##\hfil
177 \quad\hfil$##\hfil\crcr
```

```

178 \omit\strut\hfil\crcr\noalign{\kern-\baselineskip}%
179 #1\crcr\omit\strut\cr}%
180 \setbox\tw@{\vbox{\unvcopy\z@\global\setbox\one\lastbox}%
181 \setbox\tw@{\hbox{\unhbox\one\unskip\global\setbox\one\lastbox}%
182 \setbox\tw@{\hbox{$\kern\wd\one\kern-\tempdima\left(\kern-\wd\one
183 \global\setbox\one\vbox{\box\one\kern2\p@}%
184 \vcenter{\kern-\ht\one\unvbox\z@\kern-\baselineskip}\,,\right)$}%
185 \null\; \vbox{\kern\ht\one\box\tw@}\endgroup}
```

(End definition for `\bordermatrix`.)

### `\openup`

```

186 \def\openup{\afterassignment\openup\dimen@}
187 \def\openup{\advance\lineskip\dimen@%
188 \advance\baselineskip\dimen@%
189 \advance\lineskiplimit\dimen@}
```

(End definition for `\openup`.)

### `\displaylines`

```

190 \newif\ifdt@p
191 \def\displ@y{\global\dt@ptrue\openup\jot\m@th
192 \everycr{\noalign{\ifdt@p \global\dt@pfalse \ifdim\prevdepth>-1000\p@
193 \vskip-\lineskiplimit \vskip\normalineskiplimit \fi
194 \else \penalty\interdisplaylinepenalty \fi}}}
195 \def\@lign{\tabskip\z@skip\everycr{}% restore inside \displ@y
196 \def\displaylines{\displ@y \tabskip\z@skip
197 \halign{\hb@xt@\displaywidth{$\@lign\hfil\displaystyle##\hfil$}\crcr
198 #1\crcr}}
```

(End definition for `\displaylines`.)

### `\sp`

```

\sb 199 \let\sp=^
200 \let\sb=_
```

(End definition for `\sp` and `\sb`.)

### `\tmspace`

```

\thinspace
\thickspace
\negthinspace
\negthickspace
```

Originally L<sup>A</sup>T<sub>E</sub>X only provided a small set of spacing commands for use in text and math, some of the commands like `\;` were only supported in math mode. `amsmath` normalized and provided all of them in text and math. This code has now been moved to the kernel so that it is generally available.

```

\medspace
\negmedspace
\thickspace
\negthickspace
```

Originally L<sup>A</sup>T<sub>E</sub>X only provided a small set of spacing commands for use in text and math, some of the commands like `\;` were only supported in math mode. `amsmath` normalized and provided all of them in text and math. This code has now been moved to the kernel so that it is generally available.

```

201 </2ekernel>
202 <*2ekernel | latexrelease>
203 <latexrelease>\IncludeInRelease{2020/10/01}%
204 <latexrelease> {\tmspace}{amsmath spacing commands}%
```

`\tmspace` is really meant to be an internal command so it doesn't necessarily has to be robust but it was robust in `amsmath` so we leave it like that.

```

205 \DeclareRobustCommand\tmspace[3]{%
206 \ifmmode\mskip#1#2\else\leavevmode\kern#1#3\fi\relax}
```

In `amsmath` the text kern is `.16667em`. For compatibility reasons we keep the longer one.

```
207 \DeclareRobustCommand\,{\tmspace+\thinmuskip{.16667em}}
208 \let\thinspace\,
209 \DeclareRobustCommand\!{\tmspace-\thinmuskip{.16667em}}
210 \let\negthinspace\!
211 \DeclareRobustCommand\:{\tmspace+\medmuskip{.2222em}}
212 \let\medspace\:
```

LATEX has a second name for this in its manual:

```
213 \let\>=\
214 \DeclareRobustCommand\negmedspace{\tmspace-\medmuskip{.2222em}}
215 \DeclareRobustCommand\;{\tmspace+\thickmuskip{.2777em}}
216 \let\thickspace\
217 \DeclareRobustCommand\negthickspace{\tmspace-\thickmuskip{.2777em}}
218 {/2ekernel | latexrelease}
219 {/latexrelease}\EndIncludeInRelease
220 {/latexrelease}\IncludeInRelease{0000/00/00}%
221 {/latexrelease} {\tmspace}{amsmath spacing commands}%
222 {/latexrelease}
223 {/latexrelease}\let\tmspace\@undefined
224 {/latexrelease}\DeclareRobustCommand\,{\,}%
225 {/latexrelease} \relax\ifmmode\mskip\thinmuskip\else\thinspace\fi}
226 {/latexrelease}\DeclareRobustCommand\thinspace{\leavevmode@ifvmode\kern .16667em }
227 {/latexrelease}\DeclareRobustCommand\negthinspace{\leavevmode@ifvmode\kern-.16667em }
228 {/latexrelease}\def\>{\mskip\medmuskip}
229 {/latexrelease}\let\:=\>
230 {/latexrelease}\def\;{\mskip\thickmuskip}
231 {/latexrelease}\def\!{\mskip-\thinmuskip}
232 {/latexrelease}
233 {/latexrelease}\let\negmedspace\@undefined
234 {/latexrelease}\let\negthickspace\@undefined
235 {/latexrelease}
236 {/latexrelease}\EndIncludeInRelease
237 {/2ekernel}
```

(*End definition for `\tmspace` and others.*)

\\*

```
238 \DeclareRobustCommand*{\discretionary{\thinspace\the\textfont2\char2}{}{}}
```

(*End definition for \\*.*)

\!: Nickname for the medium space since `\>` is not available inside `tabbing`.

```
239 \%let\:=\>
```

(*End definition for \::*)

`\active@math@prime` This is the definition of the active math prime.

```
240 \def\active@math@prime{\^bgroup\prim@s}
```

(*End definition for \active@math@prime.*)

```

\prime@s
241 {\catcode`'=active \global\let'\active@math@prime}
242 \def\prime@s{%
243 \prime\futurelet@\let@token\prime@s}
244 \def\prime@s{%
245 \ifx`\@let@token
246 \expandafter\prime@s
247 \else
248 \ifx^`\@let@token
249 \expandafter\expandafter\expandafter\prime@ct
250 \else
251 \egroup
252 \fi
253 \fi}
254 \def\prime@s{\prime@s}
255 \def\prime@t{\prime@s\egroup}

(End definition for \prime@s.)
```

256 {\catcode`\\_=active \gdef\_`\_{\prime@t} % \_ in math is
257 % either subscript or \\_

## 1.2 Math Environments

- \(\backslash(\) Produces \$...\$ with checks that \(\backslash(\) isn't used in math mode, and that \(\backslash)\) is only used in math mode begun with \(\backslash(\).

```

258 {/2ekernel}
259 <latexrelease>\IncludeInRelease{2015/01/01}{\{}{\Make \(\backslash(\ robust)}%
260 <*2ekernel | latexrelease>
261 \DeclareRobustCommand\(\{%
262 \relax\ifmmode\@badmath\else$\fi\}%
263 \}\DeclareRobustCommand\)\{%
264 \relax\ifmmode\ifinner$\else\@badmath\fi\else \@badmath\fi\}%
265 </2ekernel | latexrelease>
266 <latexrelease>\EndIncludeInRelease
267 <latexrelease>\IncludeInRelease{0000/00/00}{\{}{\Make \(\backslash(\ robust)}%
268 <latexrelease>\def\(\{%
269 <latexrelease> \relax\ifmmode\@badmath\else$\fi\}%
270 <latexrelease>\expandafter\let\csname\string(\ \endcsname\@undefined
271 <latexrelease>\def\)\{%
272 <latexrelease> \relax\ifmmode\ifinner$\else\@badmath\fi\else \@badmath\fi\}%
273 <latexrelease>\expandafter\let\csname\string)\ \endcsname\@undefined
274 <latexrelease>\EndIncludeInRelease
275 <*2ekernel>
```

(End definition for \(\backslash(\) and \(\backslash)\).

- \(\backslash[\) Produces \$\$...\$\$ with checks that \(\backslash[\) isn't used in math mode, and that \(\backslash]\) is only used in display math mode (though there is no real test that this display math started with \(\backslash[\) and not with \$\$).

```

276 {/2ekernel}
277 <latexrelease>\IncludeInRelease{2015/01/01}{\{}{\Make \(\backslash[\ robust)}%
```

```

278 {*2ekernel | latexrelease}
279 \DeclareRobustCommand\[{%
280 \relax\ifmmode
281 \@badmath
282 \else
283 \ifvmode
284 \nointerlineskip
285 \makebox[.6\linewidth]{}%
286 \fi
287 $$$%$$$ BRACE MATCH HACK
288 \fi
289 }%
290 \DeclareRobustCommand\]{%
291 \relax\ifmmode
292 \ifinner
293 \@badmath
294 \else
295 $$$%$$$ BRACE MATCH HACK
296 \fi
297 \else
298 \@badmath
299 \fi
300 \ignorespaces
301 }%
302 //2ekernel | latexrelease)
303 <latexrelease>\EndIncludeInRelease
304 <latexrelease>\IncludeInRelease{0000/00/00}{\[]}{\Make \[robust}%
305 <latexrelease>\def\[{%
306 <latexrelease> \relax\ifmmode
307 <latexrelease> \@badmath
308 <latexrelease> \else
309 <latexrelease> \ifvmode
310 <latexrelease> \nointerlineskip
311 <latexrelease> \makebox[.6\linewidth]{}%
312 <latexrelease> \fi
313 <latexrelease> $$$%$$$ BRACE MATCH HACK
314 <latexrelease> \fi
315 <latexrelease>}%
316 <latexrelease>\expandafter\let\csname\string[\endcsname\@undefined
317 <latexrelease>\def\[]{%
318 <latexrelease> \relax\ifmmode
319 <latexrelease> \ifinner
320 <latexrelease> \@badmath
321 <latexrelease> \else
322 <latexrelease> $$$%$$$ BRACE MATCH HACK
323 <latexrelease> \fi
324 <latexrelease> \else
325 <latexrelease> \@badmath
326 <latexrelease> \fi
327 <latexrelease> \ignorespaces
328 <latexrelease>}%
329 <latexrelease>\expandafter\let\csname\string] \endcsname\@undefined
330 <latexrelease>\EndIncludeInRelease

```

```

331 <*2ekernel>

(End definition for \[and \].)

math (env.) Disguises for \(\dots\) and \[\dots\].
displaymath (env.) 332 \let\math=\(
333 \let\endmath=\)
334 \def\displaymath{[]}
335 \def\enddisplaymath[]\@ignoretrue}

equation) Numbered equations, using the counter \c@equation. Note: The document style must define \theequation etc., and do the appropriate \caddtoreset. It should also redefine \c@eqnnum if another format for the equation number is desired other than the standard (...), or to move the equation numbers to the flushleft. (See comment on the \def of \c@eqnnum.)
336 \definecounter{equation}
337 \def\equation{$$\refstepcounter{equation}}
338 \def\endequation{\eqno \hbox{\c@eqnnum}$$\@ignoretrue}
(End definition for \c@equation.)

\c@eqnnum Produces the equation number for equation and eqnarray environments. The following definition is for flushright numbers; for flushleft numbers, see leqno.clo. The equation number is set in black roman type even if an eqnarray environment appears in an italic environment.
339 \def\c@eqnnum{{\normalfont \normalcolor (\theequation)}}

(End definition for \c@eqnnum.)

\stackrel A disguise for plain TEX's buildrel.
340 \DeclareRobustCommand\stackrel[2]{\mathrel{\mathop{\#2}\limits^{\#1}}}

(End definition for \stackrel.)

\frac A disguise for plain TEX's \over.
341 \DeclareRobustCommand\frac[2]{{\begingroup\#1\endgroup\over\#2} }

(End definition for \frac.)

\sqrt Add an optional argument to plain's \sqrt to give the nth root of an expression $\sqrt[n]{e}$.
\c@sqrt 342 \DeclareRobustCommand\sqrt{\c@ifnextchar[\c@sqrt\sqrtsign}
343 \def\c@sqrt[#1]{\root #1\of}

(End definition for \sqrt and \c@sqrt.)

eqnarray (env.) Here's the eqnarray environment: Default is for left-hand side of equations to be flushright. To make them flushleft, \let\c@eqnse = \hfil.
\c@eqpen 344 \newcount\c@eqcnt
\c@eqnse 345 \newcount\c@eqpen
346 \newif\c@ifeqnsw\c@eqnswtrue
347 \newskip\c@centering
348 \c@centering = 0pt plus 1000pt

```

To get a proper `\@currentlabel` we have to redefine it for the whole display. Note that we can't use `\refstepcounter` as this results in `\@currentlabel` getting restored at the wrong and thus always writing the first label to the `.aux` file.

```

349 \def\eqnarray{%
350 \stepcounter{equation}%
351 \def\@currentlabel{\p@equation\theequation}%
352 \def\@currentcounter{equation}%
353 \global\@eqnswtrue
354 \m@th
355 \global\@eqcnt\z@
356 \tabskip\@centering
357 \let\\@\eqncr
358 $$\everycr{}\halign to\displaywidth\bgroup
359 \hskip\@centering$\displaystyle\tabskip\z@skip{##}\$@\eqnsel
360 &\global\@eqcnt\@ne\hskip \tw@\arraycolsep \hfil{##}\$@\hfil
361 &\global\@eqcnt\tw@\hskip \tw@\arraycolsep
362 \$\displaystyle{##}\$@\hfil\tabskip\@centering
363 &\global\@eqcnt\thr@@ \hb@xt@\z@\bgroup\hss##\egroup
364 \tabskip\z@skip
365 \cr
366 }
367 \def\endeqnarray{%
368 \@@eqncr
369 \egroup
370 \global\advance\c@equation\m@ne
371 $$\@ignoretrue
372 }
373 \let\@eqnsel=\relax
(End definition for \@eqcnt and others.)
```

`\nonumber` Switches off equation numbering.

```
374 \def\nonumber{\global\@eqnswfalse}
```

(End definition for `\nonumber`.)

```

\@eqncr
\@xeqncr 375 \protected\def\@eqncr{%
\@yeqncr 376 {\ifnum0='}\fi
377 \@ifstar{%
378 \global\@eqpen\@M\@yeqncr
379 }{%
380 \global\@eqpen\interdisplaylinepenalty \@yeqncr
381 }%
382 }
383 \def\@yeqncr{\@testopt\@xeqncr\z@skip}

384 </2ekernel>
385 <*2ekernel | latexrelease>
386 <latexrelease>\IncludeInRelease{2020/10/01}%
387 <latexrelease> {\@xeqncr}{eqnarray support calc syntax}%
388 \def\@xeqncr[#1]{%
389 \ifnum0='}\fi}%

```

```

390 \@@eqnccr
391 \noalign{\penalty\eqpen\vskip\jot\@vspace@calcify{#1}}%
392 }
393 </2ekernel | latexrelease>
394 <latexrelease>\EndIncludeInRelease
395 <latexrelease>\IncludeInRelease{0000/00/00}%
396 <latexrelease> {\@xeqnccr}{eqnarray support calc syntax}%
397 <latexrelease>
398 <latexrelease>\def\@xeqnccr[#1]{%
399 <latexrelease> \ifnum0='{\fi}%
400 <latexrelease> \@@eqnccr
401 <latexrelease> \noalign{\penalty\eqpen\vskip\jot\vskip #1\relax}%
402 <latexrelease>}
403 <latexrelease>\EndIncludeInRelease
404 <*2ekernel>

```

(End definition for `\@eqnccr`, `\@xeqnccr`, and `\@yeqnccr`.)

```

\@@eqnccr
405 \def\@@eqnccr{\let\reserved@a\relax
406 \ifcase\@eqcnt \def\reserved@a{& &}\or \def\reserved@a{& }%
407 \or \def\reserved@a{&}\else
408 \let\reserved@a\empty
409 \@latex@error{Too many columns in eqnarray environment}\@ehc\fi
410 \reserved@a \if@eqnsw\@eqnnum\stepcounter{equation}\fi
411 \global\@eqnswtrue\global\@eqcnt\z@\cr}

```

(End definition for `\@@eqnccr`.)

`\eqnarray*\@seqnccr`) Here's the `eqnarray*` environment:

```

412 \let\@seqnccr=\@eqnccr
413 \@namedef{eqnarray*}{\protected\def\@eqnccr{\nonumber\@seqnccr}\eqnarray}
414 \@namedef{endeqnarray*}{\nonumber\endeqnarray}

```

(End definition for `\@seqnccr`.)

`\lefteqn` `\lefteqn{FORMULA}` typesets `FORMULA` in display math style flushleft in a box of width zero.

```

415 \def\lefteqn#1{\rlap{$\displaystyle #1$}}

```

(End definition for `\lefteqn`.)

`\ensuremath` In math mode, `\ensuremath{text}` is equivalent to `text`; in LR or paragraph mode, it is equivalent to `$text$`. `\relax` is not needed in front of the `\ifmmode` as `\protect` will be `\let` to `\relax`. This version (due to Donald Arseneau) avoids duplicating its argument in the ‘then’ and ‘else’ part of the `\ifmath` which is necessary in nested ‘tabular’ like environments. See amslatex/2104.

```

416 \DeclareRobustCommand{\ensuremath}{%
417 \ifmmode
418 \expandafter\@firstofone
419 \else
420 \expandafter\@ensuredmath
421 \fi}

```

(End definition for `\ensuremath`.)

`\@ensuredmath` The `\relax` stops `\ensuremath{}` starting display math.

422 `\long\def\@ensuredmath#1{$\relax#1$}`

(End definition for `\@ensuredmath`.)

423 `\langle/2ekernel\rangle`

## 1.3 External options to the standard document classes

### 1.3.1 Left equation numbering

`\@eqnnum` To put the equation number on the left side of an equation we have to use a little trick. The number is shifted `\displaywidth` to the left inside a box of (approximately) zero width. This fails when the equation is too wide, the equation number than may overprint the equation itself.

424 `\begin{*leqno}`  
425 `\renewcommand{\@eqnnum}{\hb@xt@.01\p@{}%`  
426 `\rlap{\normalfont\color`  
427 `\hskip -\displaywidth(\theequation)}`  
428 `\end{*leqno}`

(End definition for `\@eqnnum`.)

### 1.3.2 Flush left equations

To get the displayed math environments to print the contents flush left (with an indentation) we have to redefine all of L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>'s displayed math environments.

`\mathindent` The amount of indentation of the equations is stored in a register.

429 `\begin{*fleqn}`  
430 `\newskip\mathindent`

The setting of `\mathindent` has to be deferred until the class file has been processed, because `\leftmargini` is still 0pt wide at the moment `fleqn.clo` is read in.

431 `\AtEndOfClass{\mathindent\leftmargini}`

(End definition for `\mathindent`.)

`\[` Begin display math;

432 `\IncludeInRelease{2015/01/01}{\[\]{\Make\[\ robust}}%`  
433 `\DeclareRobustCommand{\[\relax`  
434 `\ifmmode\@badmath`  
435 `\else`  
436 `\begin{trivlist}}`  
437 `\@beginparpenalty\predisplaypenalty`  
438 `\@endparpenalty\postdisplaypenalty`  
439 `\item[]\leavevmode`  
440 `\hb@xt@\linewidth\bgroup \$\m@th\displaystyle %\$`  
441 `\hskip\mathindent\bgroup`  
442 `\fi}`  
443 `\EndIncludeInRelease`

```

444 \IncludeInRelease{0000/00/00}{[]}{\Make \[robust}%
445 \renewcommand[]{\relax
446 \ifmmode\@badmath
447 \else
448 \begin{trivlist}%
449 \begin{parpenalty}\predisplaypenalty
450 \end{parpenalty}\postdisplaypenalty
451 \item[]\leavevmode
452 \hb@xt@\linewidth\bgroup \$\m@th\displaystyle \$%
453 \hskip\mathindent\bgroup
454 \fi}
455 \EndIncludeInRelease

```

(End definition for \[.)

\] end display math;

```

456 \IncludeInRelease{2015/01/01}{[]}{\Make \] robust}%
457 \DeclareRobustCommand]{\relax
458 \ifmmode
459 \egroup \$\hfil\$%
460 \egroup
461 \end{trivlist}%
462 \else \@badmath
463 \fi}
464 \EndIncludeInRelease
465 \IncludeInRelease{0000/00/00}{[]}{\Make \] robust}%
466 \renewcommand[]{\relax
467 \ifmmode
468 \egroup \$\hfil\$%
469 \egroup
470 \end{trivlist}%
471 \else \@badmath
472 \fi}
473 \EndIncludeInRelease

```

(End definition for \].)

**equation (env.)** The equation environment

```

474 \renewenvironment{equation}%
475 {\begin{parpenalty}\predisplaypenalty
476 \end{parpenalty}\postdisplaypenalty
477 \refstepcounter{equation}%
478 \trivlist \item[]\leavevmode
479 \hb@xt@\linewidth\bgroup \$\m@th\$%
480 \displaystyle
481 \hskip\mathindent}%

```

Ensure that there is at least a space between formula and equation number so that they don't bump in each other.

```

482 {$.hskip .3em minus.3em\hfil \$%
483 \displaywidth\linewidth\hbox{\eqnnum}%
484 \egroup
485 \endtrivlist}

```

`eqnarray` (*env.*) The `eqnarray` environment

```
486 \renewenvironment{eqnarray}{%
487 \stepcounter{equation}%
488 \def\@currentlabel{\p@equation\theequation}%
489 \def\@currentcounter{equation}%
490 \global\@eqnswtrue\m@th
491 \global\@eqcnt\z@
492 \tabskip\mathindent
493 \let\=\@eqncr
494 \setlength\abovedisplayskip{\topsep}%
495 \ifvmode
496 \addtolength\abovedisplayskip{\partopsep}%
497 \fi}
```

When the documentclass uses a non-zero `\parskip` setting the `\topsep` might have a negative value to compensate for that. Therefore we add `\parskip` to `\abovedisplayskip`.

```
498 \addtolength\abovedisplayskip{\parskip}%
499 \setlength\belowdisplayskip{\abovedisplayskip}%
500 \setlength\belowdisplayshortskip{\abovedisplayskip}%
501 \setlength\abovedisplayshortskip{\abovedisplayskip}%
502 $$\everycr{}\halign to\linewidth{}$$
503 \bgroup
504 \hskip\@centering
505 \$\displaystyle\tabskip\z@skip{##}@\eqnsel&%
506 \global\@eqcnt\@ne \hskip \tw@\arraycolsep \hfil{##}\hfil&%
507 \global\@eqcnt\tw@ \hskip \tw@\arraycolsep
508 \$\displaystyle{##}\hfil \tabskip\@centering&%
509 \global\@eqcnt\thr@@
510 \hb@xt@\z@\bgroup\hss##\egroup\tabskip\z@skip\cr}%
511 {\@eqncr
512 \egroup
513 \global\advance\c@equation\m@ne$$\$%
514 \ignorespace
515 }
516
```

# File J

## ltlists.dtx

### 1 List, and related environments

The generic commands for creating an indented environment – `enumerate`, `itemize`, `quote`, etc – are:

```
\list{\LABEL}{{COMMANDS}} ... \endlist
```

which can be invoked by the user as the list environment. The *LABEL* argument specifies item labeling. *COMMANDS* contains commands for changing the horizontal and vertical spacing parameters.

Each item of the environment is begun by the command `\item[ITEMLABEL]` which produces an item labeled by *ITEMLABEL*. If the argument is missing, then the *LABEL* argument of the `\list` command is used as the item label.

The label is formed by putting `\makelabel{\ITEMLABEL}` in an hbox whose width is either its natural width or else `\labelwidth`, whichever is larger. The `\list` command defines `\makelabel` to have the default definition:

```
\makelabel{\ARG} == BEGIN \hfil ARG END
```

which, for a label of width less than `\labelwidth`, puts the label flushright, `\labelsep` to the left of the item's text. However, `\makelabel` can be `\let` to another command by the `\list`'s *COMMANDS* argument.

A `\usecounter{\foo}` command in the second argument causes the counter *foo* to be initialized to zero, and stepped by every `\item` command without an argument. (`\label` commands within the list refer to this counter.)

When you leave a list environment, returning either to an enclosing list or normal text mode, LaTeX begins a new paragraph if and only if you leave a blank line after the `\end` command. This is accomplished by the `\@endparenv` command.

Blank lines are ignored every other reasonable place—i.e.:

- Between the `\begin{list}` and the first `\item`,
- Between the `\item` and the text of that item.
- Between the end of the last item and the `\end{list}`.

For an environment like quotation, in which items are not labeled, the entire environment is a single item. It is defined by letting `\quotation == \list{}{\relax}`. (Note the `\relax`, there in case the first character in the environment is a '['.) The spacing parameters provide a great deal of flexibility in designing the format, including the ability to let the indentation of the first paragraph be different from that of the subsequent ones.

The trivlist environment is equivalent to a list environment whose second argument sets the following parameter values:

`\leftmargin = 0`: causes no indentation of left margin

`\labelwidth = 0`: see below for precise effect this has.

`\itemindent = 0`: with a null label, makes first paragraph have no indentation. Succeeding paragraphs have `\parindent` indentation. To give first paragraph same indentation, set `\itemindent = \parindent` before the `\item[]`.

Every `\item` in a trivlist environment must have an argument—in many cases, this will be the null argument (`\item[]`). The trivlist environment is mainly used for paragraphing environments, like verbatim, in which there is no margin change. It provides the same vertical spacing as the list environment, and works reasonably well when it occurs immediately after an `\item` command in an enclosing list.

## 1.1 List and Trivlist

The following variables are used inside a list environment:

`\@totalleftmargin` The distance that the prevailing left margin is indented from the outermost left margin,

`\linewidth` The width of the current line. Must be initialized to `\hsize`.

`\@listdepth` A count for holding current list nesting depth.

`\makelabel` A macro with a single argument, used to generate the label from the argument (given or implied) of the `\item` command. Initialized to `\@mklab` by the `\list` command. This command must produce some stretch—i.e., an `\hfil`.

`\@inlabel` A switch that is false except between the time an `\item` is encountered and the time that TeX actually enters horizontal mode. Should be tested by commands that can be messed up by the list environment's use of `\everypar`.

`\box\@labels` When `\@inlabel = true`, it holds the labels to be put out by `\everypar`.

`\@noparitem` A switch set by `\list` when `\@inlabel = true`. Handles the case of a `\list` being the first thing in an item.

`\@noparlist` A switch set true for a list that begins an item. No `\topsep` space is added before or after `\item`'s such a list.

`\@newlist` Set true by `\list`, set false by the first text (by `\everypar`).

`\@noitemarg` Set true when executing an `\item` with no explicit argument. Used to save space. To save time, make two separate `\item` commands.

`\@nmbrrlist` Set true by `\usecounter` command, causes list to be numbered.

`\@listctr` \def'ed by `\usecounter` to name of counter.

`\@noskipsec` A switch set true by a sectioning command when it is creating an in-text heading with `\everypar`.

Throughout a list environment, `\hsize` is the width of the current line, measured from the outermost left margin to the outermost right margin. Environments like tabbing should use `\linewidth` instead of `\hsize`.

Here are the parameters of a list that can be set by commands in the `\list`'s COMMANDS argument. These parameters are all TeX skips or dimensions (defined by `\newskip` or `\newdimen`), so the usual TeX or L<sup>A</sup>T<sub>E</sub>X commands can be used to set them. The commands will be executed in vmode if and only if the `\list` was preceded by a `\par` (or something like an `\end{list}`), so the spacing parameters can be set according to whether the list is inside a paragraph or is its own paragraph.

## 1.2 Vertical Spacing (skips)

\topsep: Space between first item and preceding paragraph.

\partopsep: Extra space added to \topsep when environment starts a new paragraph (is called in vmode).

\itemsep: Space between successive items.

\parsep: Space between paragraphs within an item – the \parskip for this environment.

## 1.3 Penalties

\@beginparpenalty: put at the beginning of a list

\@endparpenalty: put at end of list

\@itempenalty: put between items.

## 1.4 Horizontal Spacing (dimens)

\leftmargin: space between left margin of enclosing environment (or of page if top level list) and left margin of this list. Must be nonnegative.

\rightmargin: analogous.

\listparindent: extra indentation at beginning of every paragraph of a list except the one started by the \item command. May be negative! Usually, labeled lists have \listparindent equal to zero.

\itemindent: extra indentation added right BEFORE an item label.

\labelwidth: nominal width of box that contains the label. If the natural width of the label <= \labelwidth, then the label is flushed right inside a box of width \labelwidth (with an \hfil). Otherwise, a box of the natural width is employed, which causes an indentation of the text on that line.

\labelsep: space between end of label box and text of first item.

## 1.5 Default Values

Defaults for the list environment are set as follows. First, \rightmargin, \listparindent and \itemindent are set to 0pt. Then, one of the commands \@listi, \@listii, ..., \@listvi is called, depending upon the current level of the list. The \@list ... commands should be defined by the document style. A convention that the document style should follow is to set \leftmargin to \leftmargini, ..., \leftmarginvi for the appropriate level. Items that aren't changed may be left alone, but everything that could possibly be changed must be reset. *Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more)*:

```
\list{LABEL}{COMMANDS} ==
BEGIN
 if \@listdepth > 5
 then LaTeX error: 'Too deeply nested'
 else \@listdepth :=G \@listdepth + 1
```

```

fi
\rightmargin := 0pt
\listparindent := 0pt
\itemindent := 0pt
\eval(@list \romannumeral\the\@listdepth) %% Set default values:
\@itemlabel :=L LABEL
\makelabel == \@mklab
@nmbrlist :=L false
COMMANDS

\@trivlist % commands common to \list and \trivlist

\parskip :=L \parsep
\parindent :=L \listparindent
\linewidth :=L \linewidth - \rightmargin - \leftmargin
\@totalleftmargin :=L \@totalleftmargin + \leftmargin
\parshape 1 \@totalleftmargin \linewidth
\ignorespaces % gobble space up to \item
END

\endlist == BEGIN \listdepth :=G \listdepth -1
 \endtrivlist
END

\@trivlist ==
BEGIN
if @newlist = T then \noitemerr fi
 %% This command removed for some forgotten reason.
\@topsepadd :=L \topsep
if @noskipsec then leave vertical mode fi %% Added 11 Jun 85
if vertical mode
 then \@topsepadd :=L \@topsepadd + \partopsep
 else \unskip \par % remove glue from end of last line
fi
if @inlabel = true
 then @noparitem :=L true
 @noparlist :=L true
 else @noparlist :=L false
 \@topsep :=L \@topsepadd
fi
\@topsep :=L \@topsep + \parskip %% Change 4 Sep 85
\leftskip :=L 0pt % Restore paragraphing parameters
\rightskip :=L \@rightskip
\parfillskip :=L 0pt + 1fil

NOTE: \setpar called on every \list in case \par has been
temporarily munged before the \list command.
\setpar{if @newlist = false then {\@par} fi}
\newlist :=G T
\outerparskip :=L \parskip

```

```

END

\trivlist ==
BEGIN
 \parsep := \parskip
 @nmbrlist := F
 \ctrivlist
 \labelwidth := 0
 \leftmargin := 0
 \itemindent := \parindent
 \citemlabel :=L "empty" %% added 93/12/13
 \makelabel{LABEL} == LABEL
END

\endtrivlist ==
BEGIN
 if @inlabel = T then \indent fi
 if horizontal mode then \unskip \par fi
 if @noparlist = true
 else if \lastskip > 0
 then \tempskipa := \lastskip
 \vskip - \lastskip
 \vskip \tempskipa - \outerparskip + \parskip
 fi
 \endparenv
 fi
END

\endparenv ==
BEGIN
 \addpenalty{@endparpenalty}
 \addvspace{@topsepadd}
 \endgroup %% ends the \begin command's \begingroup
 \par == BEGIN
 \restorepar
 \everypar{}
 \par
 END
 \everypar == BEGIN remove \lastbox \everypar{} END
 \begingroup %% to match the \end commands \endgroup
END

\item == BEGIN if math mode then WARNING fi
 if next char = [
 then \item
 else @noitemarg := true
 \item[@citemlabel]
END

\item[LAB] ==

```

```

BEGIN
if @noparitem = true
then @noparitem := false
 % NOTE: then clause hardly every taken,
 % so made a macro \odonoparitem
\box\@labels :=G \hbox{\hskip -\leftmargin
\box\@labels
\hskip \leftmargin }

if @minipage = false then
 \tempskipa := \lastskip
 \vskip -\lastskip
 \vskip \tempskipa + \outerparskip - \parskip
fi
else if @inlabel = true
 then \indent \par % previous item empty.
fi
if hmode then 2 \unskip's
 % To remove any space at end of prev.
 % paragraph that could cause a blank line.
\par
fi
if @newlist = T
 then if @nobreak = T % Kludge if list follows \section
 then \addvspace{\outerparskip - \parskip}
 else \addpenalty{\beginparpenalty}
 \addvspace{\topsep}
 \addvspace{\parskip} %% added 4 Sep 85
 fi
 else \addpenalty{\itempenalty}
 \addvspace{\itemsep}
 fi
 @inlabel :=G true
fi

\everypar{ @minipage :=G F
 @newlist :=G F
 if @inlabel = true
 then @inlabel :=G false
 \hskip -\parindent
 \box\@labels
 \penalty 0
 %% 3 Oct 85 - allow line break here
 \box\@labels :=G null
 fi
 \everypar{} }
@nobreak :=G false
if @noitemarg = true
then @noitemarg := false
if @nmbrlist
then \refstepcounter{\listctr}

```

```

 fi fi
 \@tempboxa :=L \hbox{\makelabel{LAB}}
 \box\@labels :=G \@labels \hskip \itemindent
 \hskip - (\labelwidth + \labelsep)
 if \wd \@tempboxa > \labelwidth
 then \box\@tempboxa
 else \hbox to \labelwidth {\unhbox\@tempboxa}
 fi
 \hskip\labelsep
 \ignorespaces %% gobble space up to text
END

```

\makelabel{LABEL} == ERROR %% default to catch lonely \item

```

\usecounter{CTR} == BEGIN @nmbrlist :=L true
 \@listctr == CTR
 \setcounter{CTR}{0}
END

```

DEFINE \dimen's and \count  
*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

```

\topskip
\partopsep 1 <*2ekernel>
\itemsep 2 \newskip\topsep
\parsep 3 \newskip\partopsep
\@topsep 4 \newskip\itemsep
\@topsepadd 5 \newskip\parsep
\outerparskip 6 \newskip\@topsep
7 \newskip\@topsepadd
8 \newskip\@outerparskip

```

(*End definition for \topskip and others.*)

```

\leftmargin
\rightmargin
\listparindent
\itemindent
\labelwidth
\labelsep
\@totallleftmargin
9 \newdimen\leftmargin
10 \newdimen\rightmargin
11 \newdimen\listparindent
12 \newdimen\itemindent
13 \newdimen\labelwidth
14 \newdimen\labelsep
15 \newdimen\linewidth
16 \newdimen\@totallleftmargin \@totallleftmargin=\z@

```

(*End definition for \leftmargin and others.*)

```

\leftmargini
\leftmarginii
17 \newdimen\leftmargini
18 \newdimen\leftmarginii
19 \newdimen\leftmarginiii
20 \newdimen\leftmarginiv
21 \newdimen\leftmarginv
22 \newdimen\leftmarginvi

(End definition for \leftmargini and others.)

\@listdepth
\@itempenalty
\@beginparpenalty
\@endparpenalty
23 \newcount\@listdepth \@listdepth=0
24 \newcount\@itempenalty
25 \newcount\@beginparpenalty
26 \newcount\@endparpenalty

(End definition for \@listdepth and others.)

\@labels
27 \newbox\@labels

(End definition for \@labels.)

\if@inlabel
\@inlabelfalse
28 \newif\if@inlabel \@inlabelfalse
\@inlabeltrue
(End definition for \if@inlabel, \@inlabelfalse, and \@inlabeltrue.)

\if@newlist
\@newlistfalse
29 \newif\if@newlist \@newlistfalse
\@newlisttrue
(End definition for \if@newlist, \@newlistfalse, and \@newlisttrue.)

\if@noperitem
\@noperitemfalse
30 \newif\if@noperitem \@noperitemfalse
\@noperitemtrue
(End definition for \if@noperitem, \@noperitemfalse, and \@noperitemtrue.)

\if@nopalrlist
\@nopalrlistfalse
31 \newif\if@nopalrlist \@nopalrlistfalse
\@nopalrlisttrue
(End definition for \if@nopalrlist, \@nopalrlistfalse, and \@nopalrlisttrue.)

\if@noitemarg
\@noitemargfalse
32 \newif\if@noitemarg \@noitemargfalse
\@noitemargtrue
(End definition for \if@noitemarg, \@noitemargfalse, and \@noitemargtrue.)

\if@newlist
\@newlistfalse
33 \newif\if@nmbrlist \@nmbrlistfalse
\@newlisttrue
(End definition for \if@newlist, \@newlistfalse, and \@newlisttrue.)

```

```

list\list)
 34 \def\list#1#2{%
 35 \ifnum \listdepth >5\relax
 36 \toodeep
 37 \else
 38 \global\advance\listdepth\one
 39 \fi
 40 \rightmargin\z@%
 41 \listparindent\z@%
 42 \itemindent\z@%
 43 \csname @list\romannumeral\the\listdepth\endcsname
 44 \def\itemlabel{\#1}%
 45 \let\makelabel\mklabel
 46 \nmblistfalse
 47 #2\relax
 48 \trivlist
 49 \parskip\parsep
 50 \parindent\listparindent
 51 \advance\linewidth -\rightmargin
 52 \advance\linewidth -\leftmargin
 53 \advance\@totalleftmargin \leftmargin
 54 \parshape \one \atotalleftmargin \linewidth
 55 \ignorespaces}
(End definition for \list.)

```

```

\par@deathcycles
 56 \newcount\par@deathcycles
(End definition for \par@deathcycles.)

```

- \@trivlist** Because `\par` is sometimes made a no-op it is possible for a missing `\item` to produce a loop that does not fill memory and so never gets trapped by T<sub>E</sub>X. We thus need to trap this here by setting `\par` to count the number of times a paragraph is called with no progress being made started.

```

 57 \def\@trivlist{%
 58 \if@noskipsec \leavevmode \fi
 59 \topsepadd \topsep
 60 \ifvmode
 61 \advance\@topsepadd \partopsep
 62 \else
 63 \unskip \par
 64 \fi
 65 \if@inlabel
 66 \noparitemtrue
 67 \noparlisttrue
 68 \else
 69 \if@newlist \noitemerr \fi
 70 \noparlistfalse
 71 \topsep \topsepadd
 72 \fi
 73 \advance\@topsep \parskip
 74 \leftskip\z@skip
 75 \rightskip\@rightskip
 76 \parfillskip \flushglue

```

```

77 \par@deathcycles \z@
78 \setpar{if@newlist
79 \advance\par@deathcycles \cne
80 \ifnum \par@deathcycles >\cm
81 \noitemerr
82 {\@par}%
83 \fi
84 \else
85 {\@par}%
86 \fi}%
87 \global \cnewlisttrue
88 \outerparskip \parskip}

```

(End definition for `\@trivlist`.)

`trivlistlist`)

```

89 \def\trivlist{%
90 \parsep\parskip
91 \cnbrlistfalse
92 \trivlist
93 \labelwidth\z@
94 \leftmargin\z@
95 \itemindent\z@

```

We initialise `\@itemlabel` so that a `trivlist` with an `\item` not having an optional argument doesn't produce an error message.

```

96 \let\@itemlabel\empty
97 \def\makelabel##1{##1}

```

(End definition for `\trivlist`.)

`\endlist`

```

98 \def\endlist{%
99 \global\advance\clistdepth\mone
100 \endtrivlist}

```

(End definition for `\endlist`.)

The definition of `\trivlist` used to be in `ltspacedtx` so that other commands could be ‘let to it’. They now use `\def`.

`\endtrivlist`

```

101 \def\endtrivlist{%
102 \if@inlabel
103 \leavevmode
104 \global \cinnlabelfalse
105 \fi
106 \if@newlist
107 \noitemerr
108 \global \cnewlistfalse
109 \fi
110 \ifhmode\unskip \par

```

We also check if we are in math mode and issue an error message if so (hoping that `\currenvir` resolves suitably). Otherwise the usual “perhaps a missing item” error will get triggered later which is confusing.

```

111 \else

```

```

112 \@inmatherr{\end{@currenvir}}%
113 \fi
114 \if@nopalst \else
115 \ifdim\lastskip >\z@
116 \tempskipa\lastskip \vskip -\lastskip
117 \advance\tempskipa\parskip \advance\tempskipa -\outerparskip
118 \vskip\tempskipa
119 \fi
120 \endparenv
121 \fi
122 }

```

(End definition for `\endtrivlist`.)

`\@endparenv` To suppress the paragraph indentation in text immediately following a paragraph-making environment, `\everypar` is changed to remove the space, and `\par` is redefined to restore `\everypar`. Instead of redefining `\par` and `\everypar`, `\@endparenv` was changed to set the `@endpe` switch, letting `\end` redefine `\par` and `\everypar`.

This allows paragraph-making environments to work right when called by other environments. (Changed 27 Oct 86)

```

123 \def\@endparenv{%
124 \addpenalty\@endparpenalty\addvspace\@topsepadd\@endpetrue}
125 \if@latexrelease\IncludeInRelease{2015/01/01}{\@doendpe}{clubpenalty fix}%
126 \def\@doendpe{\@endpetrue
127 \def\par{\@restorepar}

```

If a section heading changes `\clubpenalty` to keep lines after it together then this modification is restored via the `\everypar` mechanism at the start of the next paragraph. As we destroy the contents of this token here we explicitly set `\clubpenalty` back to its default.

```

128 \clubpenalty\@clubpenalty
129 \everypar{}{\par\@endpefalse}\everypar

```

Use `\setbox0=\lastbox` instead of `\hskip -\parindent` so that a `\noindent` becomes a no-op when used before a line immediately following a list environment(23 Oct 86).

```

130 \setbox\z@\lastbox}%
131 \everypar{}{\@endpefalse}%
132 \if@latexrelease\EndIncludeInRelease
133 \if@latexrelease\IncludeInRelease{0000/00/00}{\@doendpe}{clubpenalty fix}%
134 \if@latexrelease\def\@doendpe{\@endpetrue
135 \if@latexrelease\def\par{\@restorepar\everypar{}{\par\@endpefalse}\everypar
136 \if@latexrelease\setbox\z@\lastbox\everypar{}{\@endpefalse}%
137 \if@latexrelease\EndIncludeInRelease

```

(End definition for `\@endparenv` and `\@doendpe`.)

```

\if@endpe
\@endpefalse
138 \newif\if@endpe
139 \@endpefalse

```

(End definition for `\if@endpe`, `\@endpefalse`, and `\@endpeltrue`.)

```

\@mklab
140 \def\@mklab#1{\hfil #1}
(End definition for \@mklab.)

\item
141 \def\item{%
142 \cinmatherr\item
143 \ifnextchar [\@item{\@noitemargtrue \@item[\@itemlabel]}}}
(End definition for \item.)

\@donoparitem
144 \def\@donoparitem{%
145 \noparitemfalse
146 \global\setbox\@labels\hbox{\hskip -\leftmargin
147 \unhbox\@labels
148 \hskip \leftmargin}%
149 \if@minipage\else
150 \tempskipa\lastskip
151 \vskip -\lastskip
152 \advance\tempskipa\outerparskip
153 \advance\tempskipa -\parskip
154 \vskip\tempskipa
155 \fi}
(End definition for \@donoparitem.)

\@item
156 \def\@item[#1]{%
157 \if\noparitem
158 \@donoparitem
159 \else
160 \if\inlabel
161 \indent \par
162 \fi
163 \ifhmode
164 \unskip\unskip \par
165 \fi
166 \if\newlist
167 \if\nobreak
168 \nbitem
169 \else
170 \addpenalty\beginparpenalty
171 \addvspace\topsep
172 \addvspace{-\parskip}%
173 \fi
174 \else
175 \addpenalty\itempenalty
176 \addvspace\itemsep
177 \fi
178 \global\inlabeltrue
179 \fi
180 \everypar{%
181 \minipagetrue
182 \global\newlistfalse

```

This `\if@inlabel` check is needed in case an item starts of inside a group so that `\everypar` does not become empty outside that group.

```
183 \if@inlabel
184 \global\@inlabelfalse
```

The paragraph indent is now removed by using `\setbox...` since this makes `\noindent` a no-op here, as it should be. Thus the following comment is redundant but is left here for the sake of future historians: this next command was changed from an `hskip` to a `kern` to avoid a break point after the parindent box: the skip could cause a line-break if a very long label occurs in `raggedright` setting. If `\noindent` was used after `\item` want to cancel the `\itemindent` skip. This case can be detected as the indentation box will be void.

```
185 {\setbox\z@\lastbox
186 \ifvoid\z@
187 \kern-\itemindent
188 \fi}%
189 \box@\labels
190 \penalty\z@
191 \fi
```

This code is intended to prevent a page break after the first line of an item that comes immediately after a section title. It may be sensible to always forbid a page break after one line of an item? As with all such settings of `\clubpenalty` it is local so will have no effect if the item starts in a group.

Only resetting `\nobreak` when it is true is now essential since now it is sometimes set locally.

```
192 \if@nobreak
193 \nobreakfalse
194 \clubpenalty \zM
195 \else
196 \clubpenalty \clubpenalty
197 \everypar{}%
198 \fi}%

199 \if@noitemarg
200 \noitemargfalse
201 \if@nmbrlist
202 \refstepcounter\listctr
203 \fi
204 \fi
```

We use `\sbox` to support colour commands.

```
205 \sbox\@tempboxa{\makelabel{#1}}%
206 \global\setbox\@labels\hbox{%
207 \unhbox\@labels
208 \hskip \itemindent
209 \hskip -\labelwidth
210 \hskip -\labelsep
211 \ifdim \wd\@tempboxa >\labelwidth
212 \box\@tempboxa
```

```

213 \else
214 \hbox to\labelwidth {\unhbox\@tempboxa}%
215 \fi
216 \hskip \labelsep\%
217 \ignorespaces}

```

(End definition for `\@item`.)

```

\makelabel
218 \def\makelabel#1{%
219 \@latex@error{Lonely \string\item--perhaps a missing
220 list environment}\@ehc}

```

(End definition for `\makelabel`.)

```

\@nbitem
221 \def\@nbitem{%
222 \@tempskipa\@outerparskip
223 \advance\@tempskipa -\parskip
224 \addvspace\@tempskipa}

```

(End definition for `\@nbitem`.)

```

\usecounter
225 \def\usecounter#1{\@nmbrlisttrue\def\@listctr{#1}\setcounter{#1}\z@}

```

(End definition for `\usecounter`.)

## 1.6 Itemize and Enumerate

Enumeration is done with four counters: `enumi`, `enumii`, `enumiii` and `enumiv`, where `enumN` controls the numbering of the Nth level enumeration. The label is generated by the commands `\labelenumi` ... `\labelenumiv`, which should be defined by the document style. Note that `\p@enumN\theenumN` defines the output of a `\ref` command. A typical definition might be:

```

\def\theenumii{\alph{enumii}}
\def\p@enumii{\theenumii}
\def\labelenumii{(\theenumii)}

```

which will print the labels as ‘(a)’, ‘(b)’, ... and print a `\ref` as ‘3a’.

The item numbers are moved to the right of the label box, so they are always a distance of `\labelsep` from the item.

`\@enumdepth` holds the current enumeration nesting depth.

Itemization is controlled by four commands: `\labelitemi`, `\labelitemii`, `\labelitemiii`, and `\labelitemiv`. To cause the second-level list to be bulleted, you just define `\labelitemii` to be `•`. `\@itemspacing` and `\@itemdepth` are the analogs of `\@listspacing` and `\@listdepth`.

*Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*

```

\enumerate ==
BEGIN
if \@enumdepth > 3
 then errormessage: "Too deeply nested".
 else \@enumdepth :=L \@enumdepth + 1

```

```

 \c@enumctr :=L eval(enum@\romannumeral\the\c@enumdepth)
 \list{\label(\c@enumctr)}
 {\usecounter{\c@enumctr}
 \makelabel{LABEL} == \hss \llap{LABEL}}
 fi
END

```

\endenumerate == \endlist  
*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

```

\c@enumdepth
226 \newcount\c@enumdepth \c@enumdepth = 0
(End definition for \c@enumdepth.)

```

```

\c@enumi
\c@enumii 227 \c@definecounter{enumi}
\c@enumiii 228 \c@definecounter{enumii}
\c@enumiv 229 \c@definecounter{enumiii}
230 \c@definecounter{enumiv}

```

(End definition for \c@enumi and others.)

```

enumerate (env.)
231 \def\enumerate{%
232 \ifnum \c@enumdepth > \thr@@\c@toodeep\else
233 \advance\c@enumdepth\@ne
234 \edef\c@enumctr{enum\romannumeral\the\c@enumdepth}%
235 \expandafter
236 \list
237 \csname label\c@enumctr\endcsname
238 {\usecounter{\c@enumctr}\def\makelabel##1{\hss\llap{##1}}\%}
239 \fi}
240 \let\endenumerate =\endlist

```

*Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*

```

\itemize ==
BEGIN
 if \c@itemdepth > 3
 then errormessage: 'Too deeply nested'.
 else \c@itemdepth :=L \c@itemdepth + 1
 \c@itemitem == eval(labelitem\romannumeral\the\c@itemdepth)
 \list{\c@nameuse{\c@itemitem}}
 {\makelabel{LABEL} == \hss \llap{LABEL}}
 fi
END

```

\enditemize == \endlist

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

```

\@itemdepth
241 \newcount\@itemdepth \@itemdepth = 0
(End definition for \@itemdepth.)
```

**itemize** (*env.*)

```

242 \def\itemize{%
243 \ifnum \@itemdepth >\thr@@\@toodeep\else
244 \advance\@itemdepth\@ne
245 \edef\@itemitem{\labelitem\romannumeral\the\@itemdepth}%
246 \expandafter
247 \list
248 \csname\@itemitem\endcsname
249 {\def\makelabel##1{\hss\llap{##1}}}\%
250 \fi}
251 \let\enditemize =\endlist
252 ⟨/2ekernel⟩
```

# File K

## ltboxes.dtx

### 1 L<sup>A</sup>T<sub>E</sub>X Box commands

\makebox \makebox[⟨wid⟩][⟨pos⟩]{⟨obj⟩}  
Puts ⟨obj⟩ in an \hbox of width ⟨wid⟩, positioned by ⟨pos⟩.  
The possible ⟨pos⟩ are:  
**s** stretched,  
**l** flushleft,  
**r** flushright,  
**c** (default) centred.  
If ⟨wid⟩ is missing, then ⟨pos⟩ is also missing and ⟨obj⟩ is put in an \hbox of its natural width.

\makebox⟨x⟩,⟨y⟩[⟨pos⟩]{⟨obj⟩}  
Puts ⟨obj⟩ in an \hbox of width  $x * \unitlength$  and height  $y * \unitlength$ .  
⟨pos⟩ arguments are **s**, **l**, **r** or **c** (default) for stretched, flushleft, flushright or centred, and **t** or **b** for top, bottom – or combinations like **tr** or **rb**. Default for horizontal and vertical are centered. Note that in this picture mode version of \makebox a [b] aligns on the *bottom* of the text as documented. If you want to align on the *baseline* use \makebox( , )[b]{\raisebox{0pt}[\height][0pt]{xyz}} or \makebox( , )[b]{\smash{xyz}}

\mbox \mbox{⟨obj⟩} The same as \makebox{⟨obj⟩}, but is more efficient as no checking for optional arguments is done.

\newsavebox \newsavebox{⟨cmd⟩} : If \cmd is undefined, then defines it to be a T<sub>E</sub>X box register.

\savebox \savebox{⟨cmd⟩} ... : \cmd is defined to be a T<sub>E</sub>X box register, and the '...' are any \makebox arguments. It is like \makebox, except it doesn't produce text but saves the value in \box \cmd.

\sbox \sbox{⟨cmd⟩}{⟨obj⟩} is an efficient abbreviation for \savebox{⟨cmd⟩}{⟨obj⟩}.

\lrbox (env.) \begin{lrbox}{⟨cmd⟩}{⟨text⟩}\end{lrbox} is equivalent to \sbox{⟨cmd⟩}{⟨text⟩} except that any white space at the beginning and end of ⟨text⟩ is ignored.

\framebox \framebox ... : like \makebox, except it puts a 'frame' around the box. The frame is made of lines of thickness \fboxrule, separated by space \fboxsep from the text – except for \framebox(X,Y) ... , where the thickness of the lines is as for the picture environment, and there is no separation added.

\fbox \fbox{⟨obj⟩} is an abbreviation for \framebox{⟨obj⟩}.

\parbox \parbox[⟨pos⟩][⟨height⟩][⟨inner-pos⟩]{⟨width⟩}{⟨text⟩} : Makes a box with \hsize ⟨width⟩, positioned by ⟨pos⟩ as follows: c : \vcenter (placed in \$...\$ if not in math mode) b : \vbox t : \vtop default value is c. Sets \hsize := ⟨width⟩ and calls \parboxrestore, which does the following: Restores the original definitions of:

\par  
\  
\-  
\'  
\`  
\=

Resets the following parameters:

|                   |   |                     |
|-------------------|---|---------------------|
| \parindent        | = | opt                 |
| \parskip          | = | opt                 |
| \ linewidth       | = | \hsize              |
| \@totalleftmargin | = | opt                 |
| \leftskip         | = | opt                 |
| \rightskip        | = | opt                 |
| \@rightskip       | = | opt                 |
| \parfillskip      | = | opt plus 1fil       |
| \lineskip         | = | \normallineskip     |
| \baselineskip     | = | \normalbaselineskip |

Calls \sloppy  
Note: \arrayparboxrestore same as \parboxrestore but it doesn't restore \\.

**minipage (env.)**      minipage : Similar to \parbox, except it also makes this look like a page by setting \textwidth == \columnwidth == box width  
changes footnotes by redefining:  
\@mpfn == mpfootnote  
\thempfn == \thempfootnote  
\@footnotetext == \@mpfootnotetext  
resets the following list environment parameters:  
\@listdepth == \@mplistdepth  
where \@mplistdepth is initialized to zero,  
and executes \minipagerestore to allow the document style to reset any other parameters it desires. It sets @minipage true, and resets \everypar to set it false. This switch keeps \addvspace from putting space at the top of a minipage.  
Change added 24 May 89: \minipage sets @minipage globally; \endminipage resets it false.

|            |                                                                                                                                                                                                                                                                                                                                                                   |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \rule      | \rule[<raised>]{<width>}{<height>} : Makes a <width>*<height> rule, raised <raised>.                                                                                                                                                                                                                                                                              |
| \underline | \underline{<text>} : Makes an underlined hbox with <text> in it.                                                                                                                                                                                                                                                                                                  |
| \raisebox  | \raisebox{<distance>}{<height>}[<depth>]{<box>} :<br>Raises <box> up by <distance> length (down if <distance> negative). Makes T <sub>E</sub> X think that the new box extends <height> above the line and <depth> below, for a total vertical length of <height>+<depth>. Default values of <height> & <depth> = actual height and depth of box in new position. |

```

1 (*2ekernel)
2 \message{boxes,}

```

**\makebox** \makebox User level command just looks for optional [ or (.

```

3
```

- 4 (latexrelease)\IncludeInRelease{2015/01/01}%
- 5 (latexrelease) {\makebox}{Make \makebox robust}%
- 6 (\*2ekernel | latexrelease)
- 7 \DeclareRobustCommand\makebox{%
- 8 \leavevmode
- 9 \@ifnextchar(%)
- 10 \makepicbox
- 11 {\@ifnextchar[\@makebox\mbox}%
- 12 (/2ekernel | latexrelease)
- 13 (latexrelease)\EndIncludeInRelease
- 14 (latexrelease)\IncludeInRelease{0000/00/00}%
- 15 (latexrelease) {\makebox}{Make \makebox robust}%

```

16 <latexrelease>\def\makebox{%
17 <latexrelease> \leavevmode
18 <latexrelease> \@ifnextchar(%)
19 <latexrelease> \makepicbox
20 <latexrelease> {\ifnextchar[\@makebox\mbox}%
21 <latexrelease>\expandafter\let\csname makebox \endcsname\undefined
22 <latexrelease>\EndIncludeInRelease
23 {*2ekernel}

```

(End definition for `\makebox`.)

`\mbox` The basic horizontal box command for LATEX.

```
24 \DeclareRobustCommand\mbox[1]{\leavevmode\hbox{\#1}}
```

(End definition for `\mbox`.)

`\@makebox` Look for a possible second optional argument (defaults to `c`).

```

25 \def\@makebox[#1]{%
26 \ifnextchar [{\@imakebox[#1]}{\@imakebox[#1][c]}}

```

(End definition for `\@makebox`.)

`\@begin@tempboxa` Helper macro for supporting `\height`, `\width` etc. Grab #1 into `\@tempboxa` and measure it.

```

27 \long\def\@begin@tempboxa#1#2{%
28 \begingroup
29 \setbox\@tempboxa#1{\color@begingroup#2\color@endgroup}%
30 \def\width{\wd\@tempboxa}%
31 \def\height{\ht\@tempboxa}%
32 \def\depth{\dp\@tempboxa}%
33 \let\totalheight\ovr
34 \totalheight\height
35 \advance\totalheight\depth}

```

(End definition for `\@begin@tempboxa`.)

`\@end@tempboxa` End the group started by `\@begin@tempboxa`, so that the scope of `\height` only includes the ‘length’ argument to the user-command.

```
36 \let\@end@tempboxa\endgroup
```

(End definition for `\@end@tempboxa`.)

`\bm@c` Set up spacing.

```

37 \def\bm@c{\hss\unhbox\@tempboxa\hss}
38 \def\bm@l{\unhbox\@tempboxa\hss}\let\bm@t\bm@l
39 \def\bm@r{\hss\unhbox\@tempboxa}\let\bm@b\bm@r
40 \def\bm@s{\unhbox\@tempboxa}

```

(End definition for `\bm@c` and others.)

`\@imakebox` Internal form of `\makebox`.

```

41 \long\def\@imakebox[#1][#2]{%
42 \begin{\@tempboxa}\hbox{\#3}%
43 \setlength\@tempdima{\#1}%
44 \hb@xt@{\@tempdima}{\csname bm@\#2\endcsname}%
45 \end{\@tempboxa}}

```

(End definition for \@imakepicbox.)

\@makepicbox Picture mode form of \makebox.

```
46 \def\@makepicbox(#1,#2){%
47 \@ifnextchar[{\@imakepicbox(#1,#2)}{\@imakepicbox(#1,#2)[]}}
```

(End definition for \@makepicbox.)

\@imakepicbox picture mode version

```
48 (/2ekernel)
49 (*2ekernel | latexrelease)
50 (latexrelease)\IncludeInRelease{2020/10/01}%
51 (latexrelease) {\@imakepicbox}{default units}%
52 \long\def\@imakepicbox(#1,#2)[#3]#4{%
53 \@defaultunitsset\@tempdimc{#2}\unitlength
54 \vbox to\@tempdimc
55 {\let\mb@b\vss \let\mb@l\hss\let\mb@r\hss
56 \let\mb@t\vss
57 \otfor\reserved@a :=#3\do{%
58 \if s\reserved@a
59 \let\mb@l\relax\let\mb@r\relax
60 \else
61 \expandafter\let\csname mb@\reserved@a\endcsname\relax
62 \fi}%
63 \mb@t
64 \@defaultunitsset\@tempdimc{#1}\unitlength
65 \hb@xt@{\@tempdimc{\mb@l #4\mb@r}}%
66 \mb@b}
```

This kern ensures that a b option aligns on the bottom of the text rather than the baseline. this is the documented behaviour in the L<sup>A</sup>T<sub>E</sub>X Book. The kern is removed in compatibility mode.

```
67 \kern\z@}
68 (/2ekernel | latexrelease)

69 (latexrelease)\EndIncludeInRelease
70 (latexrelease)\IncludeInRelease{0000/00/00}%
71 (latexrelease) {\@imakepicbox}{default units}%
72 (latexrelease)\long\def\@imakepicbox(#1,#2)[#3]#4{%
73 (latexrelease) \vbox to#2\unitlength
74 (latexrelease) {\let\mb@b\vss \let\mb@l\hss\let\mb@r\hss
75 (latexrelease) \let\mb@t\vss
76 (latexrelease) \otfor\reserved@a :=#3\do{%
77 (latexrelease) \if s\reserved@a
78 (latexrelease) \let\mb@l\relax\let\mb@r\relax
79 (latexrelease) \else
80 (latexrelease) \expandafter\let\csname mb@\reserved@a\endcsname\relax
81 (latexrelease) \fi}%
82 (latexrelease) \mb@t
83 (latexrelease) \hb@xt@ #1\unitlength{\mb@l #4\mb@r}%
84 (latexrelease) \mb@b
85 (latexrelease) \kern\z@}
86 (latexrelease)\EndIncludeInRelease
87 (*2ekernel)
```

(End definition for \@imakepicbox.)

\set@color This macro is initially a no-op, but the color package will redefine it to insert a \special.  
88 \let\set@color\relax

(End definition for \set@color.)

\color@begingroup  
\color@endgroup  
\color@setgroup  
\normalcolor  
\color@hbox  
\color@vbox  
\color@endbox

In the past these macros were initially no-ops, and the color package redefined them to be \begingroup, \endgroup, \begingroup\set@color, \hbox\bgroup\color@begingroup, \color@endgroup\egroup. and *⟨set to main document color⟩* respectively.

Nowadays we always set the group already in the kernel as this makes the coding simpler.

89 ⟨/2ekernel⟩  
90 ⟨\*2ekernel | latexrelease⟩  
91 ⟨latexrelease⟩\IncludeInRelease{2021/06/01}⟨%  
92 ⟨latexrelease⟩ ⟨\color@begingroup}{color group settings}⟨%  
93 \let\color@begingroup\begingroup  
94 \def\color@endgroup{\endgraf\endgroup} % changed further in color package  
95 \def\color@setgroup{\color@begingroup} % remains untouched; only changed in a color pa  
96 \let\normalcolor\relax  
97 \def\color@hbox{\hbox\bgroup\color@begingroup}  
98 \def\color@vbox{\vbox\bgroup\color@begingroup}  
99 \def\color@endbox{\color@endgroup\egroup}  
100 ⟨/2ekernel | latexrelease⟩  
101 ⟨latexrelease⟩\EndIncludeInRelease  
102 ⟨latexrelease⟩\IncludeInRelease{0000/00/00}⟨%  
103 ⟨latexrelease⟩ ⟨\color@begingroup}{color group settings}⟨%  
104 ⟨latexrelease⟩  
105 ⟨latexrelease⟩\let\color@begingroup\relax  
106 ⟨latexrelease⟩\let\color@endgroup\relax  
107 ⟨latexrelease⟩\let\color@setgroup\relax  
108 ⟨latexrelease⟩\let\normalcolor\relax  
109 ⟨latexrelease⟩\let\color@hbox\relax  
110 ⟨latexrelease⟩\let\color@vbox\relax  
111 ⟨latexrelease⟩\let\color@endbox\relax  
112 ⟨latexrelease⟩  
113 ⟨latexrelease⟩\EndIncludeInRelease  
114 ⟨\*2ekernel⟩

(End definition for \color@begingroup and others.)

\newsavebox Allocate a new ‘savebox’.

115 \def\newsavebox#1{\@if definable{#1}{\newbox#1}}

(End definition for \newsavebox.)

\savebox Save #1 in a box register.

116 ⟨/2ekernel⟩  
117 ⟨latexrelease⟩\IncludeInRelease{2015/01/01}⟨%  
118 ⟨latexrelease⟩ ⟨\savebox}{Make \savebox robust}⟨%  
119 ⟨\*2ekernel | latexrelease⟩  
120 \DeclareRobustCommand\savebox[1]{%  
121 \@ifnextchar(%

```

122 {\@c savepicbox#1}{\@ifnextchar[{\@c savebox#1}{\sbox#1}}}}%
123 </2ekernel | latexrelease>
124 <latexrelease>\EndIncludeInRelease
125 <latexrelease>\IncludeInRelease{0000/00/00}%
126 <latexrelease> {\@c savebox}{\Make \sbox robust}%
127 <latexrelease>\def\sbox#1{%
128 <latexrelease> \@ifnextchar(%
129 <latexrelease> {\@c savepicbox#1}{\@ifnextchar[{\@c savebox#1}{\sbox#1}}}}%
130 <latexrelease>\expandafter\let\csname savebox \endcsname\@undefined
131 <latexrelease>\EndIncludeInRelease
132 {*2ekernel}

```

(End definition for `\sbox`.)

`\sbox` Save #1 in a box register.

```

133 \DeclareRobustCommand\sbox[2]{\setbox#1\hbox{%
134 \color@setgroup#2\color@endgroup}}

```

(End definition for `\sbox`.)

`\@c savebox` Look for second optional argument.

```

135 \def\@c savebox#1[#2]{%
136 \@ifnextchar [{\@c savebox#1[#2]}{\@c savebox#1[#2][c]}}

```

(End definition for `\@c savebox`.)

`\@c isavebox`

```

137 \long\def\@c isavebox#1[#2][#3][#4]{%
138 \sbox#1{\@imakebox[#2][#3][#4]}}

```

(End definition for `\@c isavebox`.)

`\@c savepicbox` Picture mode version of `\savebox`.

```

139 \def\@c savepicbox#1(#2,#3){%
140 \@ifnextchar [%]
141 {\@c savepicbox#1(#2,#3)}{\@c savepicbox#1(#2,#3)[]}}

```

(End definition for `\@c savepicbox`.)

`\@c isavepicbox` Picture mode version of `\savebox`.

```

142 \long\def\@c isavepicbox#1(#2,#3)[#4][#5]{%
143 \sbox#1{\@imakepicbox(#2,#3)[#4][#5]}}

```

(End definition for `\@c isavepicbox`.)

`\lrbox` `lrbox`: the new environment form of `\sbox`. Use `\aftergroup` tricks to enable a *local* assignment to be made to the box, in a way that it still has an effect *outside* the `lrbox` environment.

```

144 \def\lrbox#1{%
145 \edef\reserved@a{%
146 \endgroup
147 \setbox#1\hbox{%
148 \begingroup\aftergroup}%
149 \def\noexpand\currenvir{\currenvir}%
150 \def\noexpand\currenvline{\on@line}}%

```

```

151 \reserved@a
152 \endpfalse
153 \color@setgroup
154 \ignorespaces}

(End definition for \lrbox.)

\endlrbox End the lrbox environment.
155 \def\endlrbox{\unskip\color@endgroup}

(End definition for \endlrbox.)

\usebox unchanged
156 \DeclareRobustCommand\usebox[1]{\leavevmode\copy #1\relax}

(End definition for \usebox.)

\fbox The following definition of \fbox was written by Pavel Curtis (Extra space removed 14
Jan 88) RmS 92/08/24: Replaced occurrence of \@halfwidth by \@wholewidth
157 \DeclareRobustCommand\fbox[1]{%
158 \leavevmode
159 \hbox{%
160 \hskip-\@wholewidth
161 \vbox{%
162 \vskip-\@wholewidth
163 \hrule \height\@wholewidth
164 \hbox{%
165 \vrule\width\@wholewidth
166 #1%
167 \vrule\width\@wholewidth}%
168 \hrule\height\@wholewidth
169 \vskip-\@wholewidth}%
170 \hskip-\@wholewidth}}
}

(End definition for \fbox.)

\fboxrule user level parameters,
\fboxsep 171 \newdimen\fboxrule
172 \newdimen\fboxsep

(End definition for \fboxrule and \fboxsep.)

\fbox Abbreviated framed box command.
173 \DeclareRobustCommand\fbox[1]{%
174 \leavevmode
175 \setbox\@tempboxa\hbox{%
176 \color@begingroup
177 \kern\fboxsep{#1}\kern\fboxsep
178 \color@endgroup}%
179 \framebox\relax}

(End definition for \fbox.)

```

\framebox Framed version of \makebox.

```

180 {/2ekernel}
181 <{latexrelease}\IncludeInRelease{2015/01/01}%
182 <{latexrelease} {\framebox}{\Make \framebox robust}%
183 <{*2ekernel | latexrelease}
184 \DeclareRobustCommand\framebox{%
185 \@ifnextchar(%)
186 \@framepicbox{\@ifnextchar[\@framebox\fbox}{}%
187 {/2ekernel | latexrelease}
188 <{latexrelease}\EndIncludeInRelease
189 <{latexrelease}\IncludeInRelease{0000/00/00}%
190 <{latexrelease} {\framebox}{\Make \framebox robust}%
191 <{latexrelease}\def\framebox{%
192 \@ifnextchar(%)
193 \@framepicbox{\@ifnextchar[\@framebox\fbox}{}%
194 <{latexrelease}\expandafter\let\csname framebox \endcsname\@undefined
195 <{latexrelease}\EndIncludeInRelease
196 {*}2ekernel}

```

(End definition for \framebox.)

\@framebox Deal with optional arguments.

```

197 \def\@framebox[#1]{%
198 \@ifnextchar[%]
199 {\@iframebox[#1]}%
200 {\@iframebox[#1][c]}}

```

(End definition for \@framebox.)

\@iframebox The handling the optional arguments. In order to set the whole box, including the frame to the specified dimension, we first determine that dimension from the natural size of the text, #3. calculated width.

```

201 \long\def\@iframebox[#1][#2]{#3}%
202 \leavevmode
203 \begin{tempboxa}\hbox{#3}%
204 \setlength\tempdima{#1}%
205 \setbox\tempboxa\hb@xt@\tempdima
206 {\kern\fboxsep\csname bm@#2\endcsname\kern\fboxsep}%
207 \framebox{\kern\fboxrule}%
208 \end{tempboxa}

```

(End definition for \@iframebox.)

\@frameboxx Common part of \framebox and \fbox. #1 is a negative kern in the \framebox case so that the vertical rules do not add to the width of the box.

```

209 \def\@frameboxx#1{%
210 \tempdima\fboxrule
211 \advance\tempdima\fboxsep
212 \advance\tempdima\dp\tempboxa
213 \hbox{%
214 \lower\tempdima\hbox{%
215 \vbox{%
216 \hrule\height\fboxrule
217 \hbox{%

```

```

218 \vrule\@width\fboxrule
219 #1%
220 \vbox{%
221 \vskip\fboxsep
222 \box\@tempboxa
223 \vskip\fboxsep}%
224 #1%
225 \vrule\@width\fboxrule}%
226 \hrule\@height\fboxrule}%
227 }%
228 }%
229 }
```

(End definition for `\@frameb@x`.)

`\@framepicbox` Picture mode version.

```

230 \def\@framepicbox(#1,#2){%
231 \@ifnextchar[{\@ifframepicbox(#1,#2)}{\@ifframepicbox(#1,#2)[]}}
```

(End definition for `\@framepicbox`.)

`\@ifframepicbox` Picture mode version.

```

232 \long\def\@ifframepicbox(#1,#2)[#3]#4{%
233 \frame{\@imakepicbox(#1,#2)[#3]{#4}}}
```

(End definition for `\@ifframepicbox`.)

`\parbox` The main vertical-box command for L<sup>A</sup>T<sub>E</sub>X.

```

234 </2ekernel>
235 <latexrelease>\IncludeInRelease{2015/01/01}%
236 <latexrelease> {\parbox}{\Make \parbox robust}%
237 <*2ekernel | latexrelease>
238 \DeclareRobustCommand\parbox{%
239 \@ifnextchar[%]
240 \c@iparbox
241 {\c@iiparbox c\relax[s]}%
242 </2ekernel | latexrelease>
243 <latexrelease>\EndIncludeInRelease
244 <latexrelease>\IncludeInRelease{0000/00/00}%
245 <latexrelease> {\parbox}{\Make \parbox robust}%
246 <latexrelease>\def\parbox{%
247 \@ifnextchar[%]
248 \c@iparbox
249 {\c@iiparbox c\relax[s]}%
250 <latexrelease>\expandafter\let\csname parbox \endcsname\@undefined
251 <latexrelease>\EndIncludeInRelease
252 <*2ekernel>
```

(End definition for `\parbox`.)

`\c@iparbox` Optional argument handling.

```

253 \def\c@iparbox[#1]{%
254 \@ifnextchar[%]
255 {\c@iiparbox{#1}}%
256 {\c@iiparbox{#1}\relax[s]}}
```

(End definition for `\@iparbox`.)

`\@iiparbox` Optional argument handling.

```
257 \def\@iiparbox#1[#2]{%
258 \@ifnextchar[%]
259 {\@iiparbox{#1}{#2}}%
260 {\@iiparbox{#1}{#2}[#1]}}
```

(End definition for `\@iiparbox`.)

`\@iiiparbox` The internal version of `\parbox`.

```
261 \let\@parboxto\empty
262 \long\def\@iiiparbox#1#2[#3]#4#5{%
263 \leavevmode
264 \pboxswfalse
265 \setlength{\tempdima{#4}}%
266 \begin{\tempboxa}\vbox{\hsize\tempdima\parboxrestore#5\@par}%
267 \ifx\relax#2\else
268 \setlength{\tempdimb{#2}}%
269 \edef\@parboxto{to\the\tempdimb}%
270 \fi
271 \if#1b\vbox
272 \else\if #1t\vtop
273 \else\ifmmode\vcenter
274 \else\pboxswtrue \$\vcenter
275 \fi\fi\fi
276 \parboxto{\let\hss\vss\let\unhbox\unvbox
277 \csname bm\endcsname}%
278 \if\pboxsw \m@th\$ \fi
279 }
```

(End definition for `\@iiiparbox` and `\@parboxto`.)

`\@arrayparboxrestore` Restore various paragraph parameters.

The rational for allowing two normally global flags to be set locally here was stated originally by Donald Arseneau and extended by Chris Rowley. It is because these flags are only set globally to true by section commands, and these should never appear within boxes or, indeed, in any group; and they are only ever set globally to false when they are definitely true.

If anyone is unhappy with this argument then both flags should be treated as in `\setnobreak`; otherwise this command will be redundant.

```
280 </2ekernel>
281 <|latexrelease|\IncludeInRelease{2017-04-15}%
282 <|latexrelease| {\normallineskiplimit}%
283 <|latexrelease| {reset \lineskiplimit}%
284 <*2ekernel | latexrelease>
285 \def\@arrayparboxrestore{%
286 \let\ifnobreak\iffalse
287 \let\ifnoskipsec\iffalse
288 \let\par\@par
289 \let\-\@dischyp
```

Redefined accents to allow changes in font encoding

```
290 \let\'\@acci\let`\'@accii\let=\@acciii
291 \parindent\z@\parskip\z@skip
292 \everypar{}%
293 \linewidth\hsize
294 \@totalleftmargin\z@
295 \leftskip\z@skip \rightskip\z@skip \@rightskip\z@skip
296 \parfillskip\@flushglue
297 \lineskip\normallineskip
298 \lineskiplimit\normallineskiplimit
299 \baselineskip\normalbaselineskip
300 \sloppy}
301 ⟨/2ekernel | latexrelease⟩
302 ⟨latexrelease⟩\EndIncludeInRelease
303 ⟨latexrelease⟩\IncludeInRelease{0000-00-00}%
304 ⟨latexrelease⟩ \normallineskiplimit
305 ⟨latexrelease⟩ {reset \lineskiplimit}%
306 ⟨latexrelease⟩\def\@arrayparboxrestore{%
307 ⟨latexrelease⟩ \let\ifnobreak\iffalse
308 ⟨latexrelease⟩ \let\ifnoskipsec\iffalse
309 ⟨latexrelease⟩ \let\par\@@par
310 ⟨latexrelease⟩ \let\-\@dischyp
311 ⟨latexrelease⟩ \let\'\@acci\let`\'@accii\let=\@acciii
312 ⟨latexrelease⟩ \parindent\z@\parskip\z@skip
313 ⟨latexrelease⟩ \everypar{}%
314 ⟨latexrelease⟩ \linewidth\hsize
315 ⟨latexrelease⟩ \@totalleftmargin\z@
316 ⟨latexrelease⟩ \leftskip\z@skip \rightskip\z@skip \@rightskip\z@skip
317 ⟨latexrelease⟩ \parfillskip\@flushglue \lineskip\normallineskip
318 ⟨latexrelease⟩ \baselineskip\normalbaselineskip
319 ⟨latexrelease⟩ \sloppy}
320 ⟨latexrelease⟩\EndIncludeInRelease
321 {*2ekernel}
```

(End definition for `\@arrayparboxrestore`.)

`\@parboxrestore` Restore various paragraph parameters, and also `\`.`

```
322 \def\@parboxrestore{\@arrayparboxrestore\let\\\\@normalcr}
```

(End definition for `\@parboxrestore`.)

`\if@minipage` Switch that is true at the start of a minipage.

```
323 \def\@minipagefalse{\global\let\if@minipage\iffalse}
324 \def\@minipagetrue {\global\let\if@minipage\iftrue}
325 \Q@minipagefalse
```

(End definition for `\if@minipage`.)

`\minipage` Essentially an environment form of `\parbox`.

```
326 \def\minipage{%
327 \Q@ifnextchar[%]
328 \Q@iminiplate
329 {\Q@iiminiplate c\relax[s]}}}
```

(End definition for `\minipage`.)

`\@imminipage` Optional argument handling.

```
330 \def\@imminipage[#1]{%
331 \@ifnextchar[%]
332 {\@iimminipage{#1}}%
333 {\@iimminipage{#1}\relax[s]}}
```

(End definition for `\@imminipage`.)

`\@iimminipage` Optional argument handling.

```
334 \def\@iimminipage#1[#2]{%
335 \@ifnextchar[%]
336 {\@iimminipage{#1}{#2}}%
337 {\@iimminipage{#1}{#2}[#1]}}
```

(End definition for `\@iimminipage`.)

`\@iimminipage` Internal form of `minipage`.

```
338 \def\@iimminipage#1#2[#3]{#4}{%
339 \leavevmode
340 \pboxswfalse
341 \setlength{\tempdima}{#4}%
342 \def\@mpargs{{#1}{#2}{#3}{#4}}%
343 \setbox\tempboxa\vbox\bgroup
344 \color@begingroup
345 \hsize\tempdima
346 \textwidth\hsize \columnwidth\hsize
347 \parboxrestore
348 \def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@
349 \let\@footnotetext\mpfootnotetext
350 \let\@listdepth\mplistdepth \c@plistdepth\z@
351 \minipagerestore
352 \setminipage}
```

(End definition for `\@iimminipage`.)

`\@minipagerestore` Hook so that other styles can reset other commands in a minipage.

```
353 \let\@minipagerestore=\relax
```

(End definition for `\@minipagerestore`.)

`\endminipage`

```
354 \def\endminipage{%
355 \par
356 \unskip
357 \ifvoid\@mpfootins\else
358 \vskip\skip\@mpfootins
359 \normalcolor
360 \footnoterule
361 \unvbox\@mpfootins
362 \fi
363 \minipagetrue %% added 24 May 89
364 \color@endgroup
365 \egroup
366 \expandafter\@iiparbox\@mpargs{\unvbox\@tempboxa}}
```

(End definition for \endminipage.)

\@mplistdepth Versions of \clistdepth and \footins local to minipage.  
\@mpfootins  
367 \newcount\@mplistdepth  
368 \newinsert\@mpfootins  
(End definition for \@mplistdepth and \@mpfootins.)

\@mpfootnotetext Minipage version of \@footnotetext.  
Final \strut added 27 Mar 89, on suggestion by Don Hosek  
369 </2ekernel>  
370 <\*2ekernel | latexrelease>  
371 <latexrelease>\IncludeInRelease{2021/11/15}%  
372 <latexrelease> {\@mpfootnotetext}{footnotetext tagging} %  
373 \long\def\@mpfootnotetext#1{ %  
374 \global\setbox\@mpfootins\vbox{ %  
375 \unvbox\@mpfootins  
376 \reset@font\footnotesize  
377 \hsize\columnwidth  
378 \parboxrestore  
379 \def\@currentcounter{\mpfootnote} %  
380 \protected@edef\@currentlabel  
381 {\csname p@\mpfootnote\endcsname\@thefnmark} %  
382 \color@begingroup  
383 \makefntext{ %  
384 \rule{z@\footnotesep}{\ignorespaces#1\@finalstrut\strutbox}} %  
385 \par  
386 \color@endgroup } %  
387 </2ekernel | latexrelease>  
388 <latexrelease>\EndIncludeInRelease  
389 <latexrelease>\IncludeInRelease{2021/06/01}%  
390 <latexrelease> {\@mpfootnotetext}{footnotetext tagging} %  
391 <latexrelease>\long\def\@mpfootnotetext#1{ %  
392 <latexrelease> \global\setbox\@mpfootins\vbox{ %  
393 <latexrelease> \unvbox\@mpfootins  
394 <latexrelease> \reset@font\footnotesize  
395 <latexrelease> \hsize\columnwidth  
396 <latexrelease> \parboxrestore  
397 <latexrelease> \protected@edef\@currentlabel  
398 {\csname p@\mpfootnote\endcsname\@thefnmark} %  
399 <latexrelease> \color@begingroup  
400 <latexrelease> \makefntext{ %  
401 \rule{z@\footnotesep}{\ignorespaces#1\@finalstrut\strutbox}} %  
402 <latexrelease> \par  
403 <latexrelease> \color@endgroup } %  
404 <latexrelease>\EndIncludeInRelease  
405 <latexrelease>\IncludeInRelease{0000/00/00}%  
406 <latexrelease> {\@mpfootnotetext}{footnotetext tagging} %  
407 <latexrelease>  
408 <latexrelease>\long\def\@mpfootnotetext#1{ %  
409 <latexrelease> \global\setbox\@mpfootins\vbox{ %  
410 <latexrelease> \unvbox\@mpfootins  
411 <latexrelease> \reset@font\footnotesize

```

412 <|latexrelease> \hsize\columnwidth
413 <|latexrelease> \parboxrestore
414 <|latexrelease> \protected@edef\@currentlabel
415 {\csname p@mpfootnote\endcsname\@thefnmark}%
416 <|latexrelease> \color@begingroup
417 <|latexrelease> \makefntext{%
418 \rule{z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
419 <|latexrelease> \color@endgroup}%
420 <|latexrelease>
421 <|latexrelease>\EndIncludeInRelease
422 (*2ekernel)

```

(*End definition for \@mpfootnotetext.*)

```
423 \newif\if@pboxsw
```

**\rule** Draw a rule of the specified size.

```

424 </2ekernel>
425 <|latexrelease>\IncludeInRelease{2015/01/01}%
426 <|latexrelease> {\rule}{\Make \rule robust}%
427 (*2ekernel | latexrelease)
428 \DeclareRobustCommand\rule{\@ifnextchar[\@rule{\rule[\z@]}{%
429 </2ekernel | latexrelease>
430 <|latexrelease>\EndIncludeInRelease
431 <|latexrelease>\IncludeInRelease{0000/00/00}%
432 <|latexrelease> {\rule}{\Make \rule robust}%
433 <|latexrelease>\def\rule{\@ifnextchar[\@rule{\rule[\z@]}{%
434 <|latexrelease>\expandafter\let\csname rule \endcsname\@undefined
435 <|latexrelease>\EndIncludeInRelease
436 (*2ekernel)

```

(*End definition for \rule.*)

**\@rule** Internal form of \rule.

```

437 \def\@rule[#1]#2#3{%
438 \leavevmode
439 \hbox{%
440 \setlength\@tempdima{#1}%
441 \setlength\@tempdimb{#2}%
442 \setlength\@tempdimc{#3}%
443 \advance\@tempdimc\@tempdima
444 \vrule\@width\@tempdimb\@height\@tempdimc\@depth-\@tempdima}%

```

(*End definition for \@rule.*)

**\@@underline** Saved primitive \underline.

```
445 \let\@@underline\underline
```

(*End definition for \@@underline.*)

**\underline** L<sup>A</sup>T<sub>E</sub>X version works outside math.

```

446 \DeclareRobustCommand\underline[1]{%
447 \relax
448 \ifmmode\@@underline{#1}%
449 \else $@\@@underline{\hbox{#1}}\m@th$\relax\fi}

```

(End definition for \underline.)

\raisebox Raise a box, and change its vertical dimensions.

```
450 </2ekernel>
451 <|latexrelease>\IncludeInRelease{2015/01/01}%
452 <|latexrelease> {\raisebox}{\Make \raisebox robust}%
453 <*2ekernel | latexrelease>
454 \DeclareRobustCommand\raisebox[1]{%
455 \leavevmode
456 \@ifnextchar[{\@rsbox{#1}}{\@irsbox{#1}[]}]
457 </2ekernel | latexrelease>
458 <|latexrelease>\EndIncludeInRelease
459 <|latexrelease>\IncludeInRelease{0000/00/00}%
460 <|latexrelease> {\raisebox}{\Make \raisebox robust}%
461 <|latexrelease>\def\raisebox#1{%
462 <|latexrelease> \leavevmode
463 <|latexrelease> \@ifnextchar[{\@rsbox{#1}}{\@irsbox{#1}[]}]
464 <|latexrelease>\expandafter\let\csname raisebox \endcsname\@undefined
465 <|latexrelease>\EndIncludeInRelease
466 <*2ekernel>
```

(End definition for \raisebox.)

\@rsbox Optional argument handling.

```
467 \def\@rsbox#1[#2]{%
468 \@ifnextchar[{\@iirsbox{#1}[#2]}{\@irsbox{#1}[#2]}}
```

(End definition for \@rsbox.)

\@argsbox ...

(End definition for \@argsbox.)

\@irsbox Internal version of \raisebox (less than two optional args).

```
469 \long\def\@irsbox#1[#2]#3{%
470 \begin{tempboxa}\hbox{#3}%
471 \setlength\tempdima{#1}%
472 \ifx\\#2\\\else\setlength\tempdimb{#2}\fi
473 \setbox\tempboxa\hbox{\raise\tempdima\box\tempboxa}%
474 \ifx\\#2\\\else\ht\tempboxa\tempdimb\fi
475 \box\tempboxa
476 \end{tempboxa}
```

(End definition for \@irsbox.)

\@iirsbox Internal version of \raisebox (two optional args).

```
477 \long\def\@iirsbox#1[#2][#3]#4{%
478 \begin{tempboxa}\hbox{#4}%
479 \setlength\tempdima{#1}%
480 \setlength\tempdimb{#2}%
481 \setlength\dimen@{#3}%
482 \setbox\tempboxa\hbox{\raise\tempdima\box\tempboxa}%
483 \ht\tempboxa\tempdimb
484 \dp\tempboxa\dimen@
485 \box\tempboxa
486 \end{tempboxa}
```

(End definition for `\@iirsbox`.)

- `\@finalstrut` This macro adds a special strut the *depth* of the box given as #1, and height and width 0pt. It is used for ensuring that the last line of a paragraph has the correct depth in ‘p’ columns of tables and in footnotes. In vertical mode nothing is done, as adding the strut (as done in 2.09) would start a new paragraph. It would be possible to inspect `\prevdepth` to check the depth of the just-completed paragraph, but we do not do that here. Actually we do even less now, skip the vmode test as it broke tabular ‘p’ columns.

The `\nobreak` was added (1995/10/31) to allow hyphenation of the final word of the paragraph.

```
487 \def\@finalstrut#1{%
488 \unskip\ifhmode\nobreak\fi\vrule\@width\z@\@height\z@\@depth\dp#1}
```

(End definition for `\@finalstrut`.)

## 1.1 Some low-level constructs

The following commands are basically inherited from plain TeX.

- `\leftline` These macros place text on a full line either centred or left or right adjusted.  
`\rightline` 489 `\def\@cline{\hb@xt@\hsize}`  
`\centerline` 490 `\DeclareRobustCommand\leftline[1]{\@cline{\#1\hss}}`  
  `\@cline` 491 `\DeclareRobustCommand\rightline[1]{\@cline{\hss\#1}}`  
  `\centerline` 492 `\DeclareRobustCommand\centerline[1]{\@cline{\hss\#1\hss}}`

(End definition for `\leftline` and others.)

- `\rlap` These macros place text to the left or right of the current reference point without taking up space.  
`\llap`

```
493 \DeclareRobustCommand\rlap[1]{\hb@xt@\z@{\#1\hss}}
494 \DeclareRobustCommand\llap[1]{\hb@xt@\z@{\hss\#1}}
```

And here is the version that centers, it was initially introduced by `mathtools`.

```
495 \DeclareRobustCommand\clap[1]{\hb@xt@\z@{\hss\#1\hss}}
```

(End definition for `\rlap`, `\llap`, and `\clap`.)

```
496 </2ekernel>
```

# File L

## lttab.dtx

### 1 Tabbing, Tabular and Array Environments

This section deals with ‘Lining It Up in Columns’. First the `tabbing` environment is defined, and then in second part, `tabular` together with its variants, `tabular*` and `array`.

Note that the `tabular` defined here is essentially the original L<sup>A</sup>T<sub>E</sub>X 2.09 version, not the extended version described in *The L<sup>A</sup>T<sub>E</sub>X Companion*. Use the `array` package to obtain the extended version.

#### 1.1 tabbing

*Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*

```
\dimen(\@firsttab + i) = distance of tab stop i from left margin
 0 <= i <= 15 (?).

\dimen\@firsttab is initialized to \totallmargin, so it starts
 at the prevailing left margin.

\@maxtab = number of highest defined tab register
 probably = \@firsttab + 12
\@nxttabmar = tab stop number of next line's left margin
\@curtabmar = tab stop number of current line's left margin
\@curtab = number of the current tab. At start of line,
 it equals \@curtabmar
\@hightab = largest tab number currently defined.
\@tabpush = depth of \pushtab's

\box\@curline = contents of current line, excluding left margin
 skip, and excluding contents of current field
\box\@curfield = contents of current field

@rjfield = switch: T iff the last field of the line should
 be right-justified at the right margin.

\tabbingsep = distance left by the \` command between the
 current position and the field that is
 “left-shifted”.

UTILITY MACROS
\@stopfield : closes the current field
\@addfield : adds the current field to the current line.
\@contfield : continues the current field
\@startfield: begins the next field
\@stopline : closes the current line and outputs it
```

```

\@startline : starts the next line
\ifatmargin : an \if that is true iff the current line.
 has width zero

\@startline ==
BEGIN
 \@curtabmar :=G \@nxttabmar
 \@curtab :=G \@curtabmar
 \box\@curline :=G null
 \@startfield
 \strut
END

\@stopline ==
BEGIN
 \unskip
 \@stopfield
 if @rjfield = T
 then @rjfield :=G F
 \tempdima := \totallmargin + \ linewidth
 \hbox@xt@ \tempdima{\itemfudge
 \hskip \dimen\@curtabmar
 \box\@curline
 \hfil
 \box\@curfield}
 else \addfield
 \hbox {\itemfudge
 \hskip \dimen\@curtabmar
 \box\@curline}
 fi
END

\@startfield ==
BEGIN
 \box\@curfield :=G \hbox {
END

\@stopfield ==
BEGIN
 }
END

\@contfield ==
BEGIN
 \box\@curfield :=G \hbox { \unhbox\@currfield %%} brace matching
END
\@addfield ==
BEGIN
 \box\@curline :=G \unbox\@curline * \unbox\@curfield
END

```

```

\@ifatmargin ==
BEGIN
 if dim of box\@curline = 0pt then
END

\tabbing ==
BEGIN
 \lineskip :=L 0pt
 \> == \@rtab
 \< == \@ltab
 \= == \@settab
 \+ == \@tabplus
 \- == \@tabminus
 \` == \@tabrj
 \' == \@tablab
 \\ == BEGIN \@stopline \@startline END
 \\[DIST] == BEGIN
 \@stopline \vskip DIST \@startline\ignorespaces END
 * == BEGIN \@stopline \penalty 10000 \@startline END
 *[DIST] == BEGIN \@stopline \penalty 10000 \vskip DIST
 \@startline\ignorespaces END
 \@heighttab := \@nxttabmar :=G \@firsttab
 \@tabpush :=G 0
 \dimen\@firsttab := \@totallleftmargin
 @rjfield :=G F
 \trivlist \item\relax
 if @minipage = F then \vskip \parskip fi
 \box\@tabbbox = \rlap{\indent\the\everypar}
 % note: \the\everypar sets @inlabel :=G F
 \@itemfudge == BEGIN \box\@tabbbox END
 \@startline
 \ignorespaces
END

\@endtabbing ==
BEGIN
 \@stopline
 if \@tabpush > 0 then error message: "unmatched \poptabs" fi
 \endtrivlist
END

\@rtab ==
BEGIN
 \@stopfield
 \@addfield
 if \@curtab < \@heighttab
 then \@curtab :=G \@curtab + 1
 else error message "Undefined Tab" fi

```

```

\@tempdima := \dimen\@curtab - \dimen\@curtabmar
 - width of box \@curline
\box\@curline :=G \hbox{\unhbox\@curline + \hskip\@tempdima}
\@startfield
END

\@settab ==
BEGIN
\@stopfield
\@addfield
if \@curtab < \@maxtab
 then \@curtab :=G \@curtab+1
 else error message: "Too many tabs" fi
if \@curtab > \@hightab
 then \@hightab :=L \@curtab fi
\dimen\@curtab :=L \dimen\@curtabmar + width of \box\@curline
\@startfield
END

\@ltab ==
BEGIN
\@ifatmargin
 then if \@curtabmar > \@firsttab
 then \@curtab :=G \@curtab - 1
 \@curtabmar :=G \@curtabmar - 1
 else error message "Too many untabs" fi
 else error message "Left tab in middle of line"
 fi
END

\@tabplus ==
BEGIN
if \@nxttabmar < \@hightab
 then \@nxttabmar :=G \@nxttabmar+1
 else error message "Undefined tab"
fi
END

\@tabminus ==
BEGIN
if \@nxttabmar > \@firsttab
 then \@nxttabmar :=G \@nxttabmar-1
 else error message "Too many untabs"
fi
END

\@tabrj ==
BEGIN \@stopfield
\@addfield
@rjfield :=G T

```

```

 \@startfield
END

\@tablab ==
BEGIN \@stopfield
 \box\@curline G:= \hbox{\box\@curline %% 'G' added 17 Jun 86
 \hskip - width of \box\@curfield
 \hskip -\tabbingsep
 \box\@curfield
 \hskip \tabbingsep }

 \@startfield
END

\pushtabs ==
BEGIN
 \@stopfield
 \tabpush :=G \tabpush + 1
 \begingroup
 \@contfield
END

\poptabs ==
BEGIN
 \@stopfield
 if \tabpush > 0
 then \endgroup
 \tabpush :=G \tabpush - 1
 else error message: "Too many \poptabs"
 fi
 \@contfield
END

```

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

- \a The accents ‘`’, ‘’’, and ‘=’ that have been redefined inside a tabbing environment can be called by typing \a‘, \a’, and \a=. The macro \a is defined in `ltoutenc.dtx`.

(*End definition for \a.*)

The ‘2ekernel’ code ensures that a \usepackage{autotabg} is essentially ignored if a ‘full’ format is being used that has picture mode already in the format.

1 \expandafter\let\csname ver@autotabg.sty\endcsname\fmtversion

```

\@firsttab
\@maxtab
 1 \expandafter\let\csname ver@autotabg.sty\endcsname\fmtversion
 2 \newdimen\@tempa
 3 \chardef\@firsttab=\the\allocationnumber
 4 \newdimen\@tempa\newdimen\@tempa\newdimen\@tempa\newdimen\@tempa
 5 \newdimen\@tempa\newdimen\@tempa\newdimen\@tempa\newdimen\@tempa
 6 \newdimen\@tempa\newdimen\@tempa\newdimen\@tempa\newdimen\@tempa
 7 \newdimen\@tempa\newdimen\@tempa\newdimen\@tempa\newdimen\@tempa
 8 \newdimen\@tempa
 9 \chardef\@maxtab=\the\allocationnumber
10 \dimen\@firsttab=0pt

```

(End definition for `\@firsttab` and `\@maxtab`.)

```
\@nxttabmar
\@curtabmar 11 \newcount\@nxttabmar
\@curtab 12 \newcount\@curtabmar
\@hightab 13 \newcount\@curtab
\@tabpush 14 \newcount\@hightab
15 \newcount\@tabpush
```

(End definition for `\@nxttabmar` and others.)

```
\@curline
\@curfield 16 \newbox\@curline
\@tabbbox 17 \newbox\@curfield
18 \newbox\@tabbbox
```

(End definition for `\@curline`, `\@curfield`, and `\@tabbbox`.)

```
\if@rjfield
19 \newif\if@rjfield
```

(End definition for `\if@rjfield`.)

`\@startline` It is, in some sense, an error if the current margin tab setting is higher than the value of `\@hightab` (which is a local variable). That this is allowed is a fundamental design flaw which is not going to be corrected now.

```
20 \def\@startline{
21 \ifnum \@nxttabmar >\@hightab
22 \@badtab
23 \global\@nxttabmar \@hightab
24 \fi
25 \global\@curtabmar \@nxttabmar
26 \global\@curtab \@curtabmar
27 \global\setbox\@curline \hbox {}%
28 \@startfield
29 \strut}
```

(End definition for `\@startline`.)

```
\@stopline
30 \def\@stopline{
31 \unskip
32 \@stopfield
33 \if@rjfield
34 \global\@rjfieldfalse
35 \tempdima\@totalleftmargin
36 \advance\tempdima\ linewidth
37 \hb@xt@\tempdima{
38 \itemfudge\hskip\dimen\@curtabmar
39 \box\@curline
40 \hfil
41 \box\@curfield} %
42 \else
43 \addfield
44 \hbox{\itemfudge\hskip\dimen\@curtabmar\box\@curline} %
45 \fi}
```

```

(End definition for \@stopline.)

\@startfield
46 \def\@startfield{%
47 \global\setbox\@curfield\hbox\bgroup\color@begingroup}
(End definition for \@startfield.)

\@stopfield
48 \def\@stopfield{%
49 \color@endgroup\egroup}
(End definition for \@stopfield.)

\@contfield
50 \def\@contfield{%
51 \global\setbox\@curfield\hbox\bgroup\color@begingroup
52 \unhbox\@curfield}
(End definition for \@contfield.)

\@addfield
53 \def\@addfield{\global\setbox\@curline\hbox{\unhbox
54 \@curline\unhbox\@curfield}}
(End definition for \@addfield.)

\@ifatmargin
55 \def\@ifatmargin{\ifdim \wd\@curline =\z@}
(End definition for \@ifatmargin.)

\@tabcr
56 \protected\def\@tabcr{\@stopline \@ifstar{\penalty \OM \@xtabcr}\@xtabcr}
(End definition for \@tabcr.)

\@xtabcr
57 \def\@xtabcr{\@ifnextchar[\@itabcr{\@startline\ignorespaces}}
(End definition for \@xtabcr.)

\@itabcr
58 </2ekernel>
59 <*2ekernel | latexrelease>
60 <latexrelease>\IncludeInRelease{2020/10/01}%
61 <latexrelease> \{@itabcr}{Tabbing calc syntax}%
62 \def\@itabcr[#1]{\@vspace@calcify[#1]\@startline\ignorespaces}
63 </2ekernel | latexrelease>
64 <latexrelease>\EndIncludeInRelease
65 <latexrelease>\IncludeInRelease{0000/00/00}%
66 <latexrelease> \{@itabcr}{Tabbing calc syntax}%
67 <latexrelease>
68 <latexrelease>\def\@itabcr[#1]{\vskip #1\@startline\ignorespaces}
69 <latexrelease>\EndIncludeInRelease
70 <*2ekernel>

```

```

tabbing (env.) We use \relax to prevent \item from scanning too far.

\tabbing 71 \def\tabbing{\lineskip \z@skip\let\>\@rtab\let\<\@ltab\let\=\@settab
72 \let\+\@tabplus\let\-\@tabminus\let\`{\@tabrj\let\'{\@tablab
73 \let\\=\@tabcr
74 \chightab\firstab
75 \global\@nxttabmar\firstab
76 \dimen\firstab\@totalleftmargin
77 \global\@tabpush\z@\global\@rjfieldfalse
78 \trivlist \item\relax
79 \if@minipage\else\vskip\parskip\fi
80 \setbox\@tabfbox\hbox{%
81 \rlap{\hskip\@totalleftmargin\indent\the\everypar}}%
82 \def\@itemfudge{\box\@tabfbox}%
83 \startline\ignorespaces}

\endtabbing 84 \def\endtabbing{%
85 \stopline\ifnum\@tabpush >\z@\badpoptabs \fi\endtrivlist}

Omitted \global added to \@rtab 17 Jun 86

\@rtab 86 \def\@rtab{\@stopfield\@addfield\ifnum \@curtab<\chightab
87 \global\advance\@curtab \one\else\badtab\fi
88 \tempdima\dimen\@curtab
89 \advance\@tempdima -\dimen\@curtabmar
90 \advance\@tempdima -\wd\curline
91 \global\setbox\curline\hbox{\unhbox\curline\hskip\@tempdima}%
92 \startfield\ignorespaces}

\@settab 93 \def\@settab{\@stopfield\@addfield
94 \ifnum \@curtab <\maxtab
95 \ifnum\@curtab =\chightab
96 \advance\chightab \one
97 \fi
98 \global\advance\@curtab \one
99 \else
100 \latexerror{Tab overflow}\ehd
101 \fi
102 \dimen\@curtab \dimen\@curtabmar
103 \advance\dimen\@curtab \wd\curline
104 \startfield
105 \ignorespaces}

\@ltab 106 \def\@ltab{\@ifatmargin\ifnum\@curtabmar >\@firstab
107 \global\advance\@curtab \m@ne \global\advance\@curtabmar \m@ne\else
108 \badtab\fi\else
109 \latexerror{\string\<\space in mid line}\ehd\fi\ignorespaces}

\@tabplus 110 \def\@tabplus{%
111 \ifnum\@nxttabmar<\chightab

```

```

112 \global\advance\@nxttabmar\@ne
113 \else
114 \@badtab
115 \fi
116 \ignorespaces}

\@tabminus 117 \def\@tabminus{%
118 \ifnum\@nxttabmar>\@firsttab
119 \global\advance\@nxttabmar\m@ne
120 \else
121 \@badtab
122 \fi
123 \ignorespaces}

\@tabrj 124 \def\@tabrj{%
125 \@stopfield\@addfield\global\@rjfieldtrue\@startfield\ignorespaces}

\setbox\@curline made \global in \@tablab. 17 Jun 86

\@tablab 126 \def\@tablab{%
127 \@stopfield
128 \global\setbox\@curline\hbox{%
129 \box\@curline
130 \hskip-\wd\@curfield \hskip-\tabbingsep
131 \box\@curfield
132 \hskip\tabbingsep}%
133 \@startfield
134 \ignorespaces}

135 </2ekernel>
136 <*2ekernel | latexrelease>
137 <latexrelease>\IncludeInRelease{2019/10/01}%
138 <latexrelease> {\pushtabs}{Make commands robust}%

\pushtabs 139 \DeclareRobustCommand\pushtabs{%
140 \@stopfield\@addfield\global\advance\@tabpush \@ne \begingroup
141 \@contfield}

```

It is, in some sense, an error if, after the endgroup, the current tab setting is higher than the new value of \chightab (which is a local variable). That this is allowed is a fundamental design flaw which is not going to be corrected now.

```

142 \DeclareRobustCommand\poptabs{\@stopfield\@addfield
143 \ifnum \@tabpush >\z@
144 \endgroup
145 \global\advance\@tabpush \m@ne
146 \ifnum \@curtab >\chightab
147 \global \@curtab \chightab
148 \@badtab
149 \fi
150 \else
151 \@badpoptabs
152 \fi
153 \@contfield}

```

```

154 \DeclareRobustCommand\kill{\@stopfield\@startline\ignorespaces}

(End definition for \@itabcr and others.)

155 </2ekernel | latexrelease>
156 <latexrelease>\EndIncludeInRelease
157 <latexrelease>\IncludeInRelease{0000/00/00}%
158 <latexrelease> {\pushtabs}{Make commands robust}%
159 <latexrelease>
160 <latexrelease>\kernel@make@fragile\pushtabs
161 <latexrelease>\kernel@make@fragile\poptabs
162 <latexrelease>\kernel@make@fragile\kill
163 <latexrelease>
164 <latexrelease>\EndIncludeInRelease
165 <*2ekernel>

\tabbingsep
166 \newdimen\tabbingsep

(End definition for \tabbingsep.)

```

## 1.2 array and tabular environments

*Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*

ARRAY PARAMETERS:

|                 |                                                                                                                                                                                                                     |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \arraycolsep    | : half the width separating columns in an array environment                                                                                                                                                         |
| \tabcolsep      | : half the width separating columns in a tabular environment                                                                                                                                                        |
| \arrayrulewidth | : width of rules                                                                                                                                                                                                    |
| \doublerulesep  | : space between adjacent rules in array or tabular                                                                                                                                                                  |
| \arraystretch   | : line spacing in array and tabular environments is done by<br>placing a strut in every row of height and depth<br>\arraystretch times the height and depth of the strut<br>produced by an ordinary \strut command. |

PREAMBLE:

The PREAMBLE argument of an array or tabular environment can contain the following:

|           |                                                                                                      |
|-----------|------------------------------------------------------------------------------------------------------|
| l,r,c     | : indicate where entry is to be placed.                                                              |
|           | : for vertical rule                                                                                  |
| @{EXP}    | : inserts the text EXP in every column.<br>\arraycolsep or \tabcolsep spacing is suppressed.         |
| *{N}{PRE} | : equivalent to writing N copies of PRE in the preamble.<br>PRE may contain *{N'}{EXP'} expressions. |
| p{LEN}    | : makes entry in parbox of width LEN.                                                                |

SPECIAL ARRAY COMMANDS:

**\multicolumn{N}{FORMAT}{ITEM}** : replaces the next N column items by ITEM, formatted according to FORMAT.  
 FORMAT should contain at most one l,r or c.  
 If it contains none, then ITEM is ignored.

**\vline** : draws a vertical line the height of the current row. May appear in an array element entry.

**\hline** : draws a horizontal line between rows. Must appear either before the first entry (to appear above the first row) or right after a \\ command. If followed by another \hline, then adds a \vskip of \doublerulesep.

**\cline{i-j}** : draws horizontal lines between rows covering columns i through j, inclusive. Multiple commands may follow one another to provide lines covering several disjoint columns

**\extracolsep{WIDTH}** : for use inside an @ in the preamble. Causes a WIDTH space to be added between columns for the rest of the columns. This is in addition to the ordinary intercolumn space.

```

\array ==
BEGIN
 \@acol == \@arrayacol
 \@classz == \@arrayclassz
 \@classiv == \@arrayclassiv
 \\ == \@arraycr
 \@halignto == NULL
 \@tabarray
END

\endarray{NAME} == BEGIN \crcr } END

\tabular ==
BEGIN
 \@halignto == NULL
 \@tabular
END

\tabular*{WIDTH} ==
BEGIN
 \@halignto == to WIDTH
 \@tabular
END

\@tabular ==
BEGIN
 \leavemode
 \hbox { $
 \@acol == \@tabacol

```

```

\@classz == \@tabclassz
\@classiv == \@tabclassiv
\\ == \@tabularcr
\@tabarray
END

\endtabular == BEGIN \crcr{} $} END

\@tabarray == if next char = [then \@array else \@array[c] fi

\@array[POS]{PREAMBLE} ==
BEGIN
 define \@arstrutbox to make \@arstrut produce strut of height
 and depth \arraystretch times the height and
 depth of a normal strut.
\@mkpream{PREAMBLE}
\@preamble == \halign \@halignto {\tabskip=0pt\@arstrut
 eval{\@preamble}\tabskip = 0pt\cr %}
\@startpbox == \@@@startpbox
\@endpbox == \@@@endpbox
if POS = t then \vtop
 else if POS = b then \vbox
 else \vcenter
 fi
 fi
{
\par ==L {} % changed 92/09/18
\@sharp == #
\protect == \relax
\lineskip :=L 0pt
\baselineskip :=L 0pt
\@preamble
END

\@arraycr ==
BEGIN
$ %% Prevents extra space at end of row's last entry.
if next char = [
 then \@argarraycr
 else $ \cr %% Needed to balance $
END

\@argarraycr[LENGTH] ==
BEGIN
$ %% Needed to balance $ of \@arraycr
if LENGTH > 0
 then \@tempdima := depth of \@arstrutbox + LENGTH
 \vrule height 0pt width 0pt depth \@tempdima
 \cr
 else \cr \noalign{\vskip LENGTH}

```

END

\@tabularcr and \@argtabularcr same as \@arraycr and \@argarraycr except without the extra \$'s.

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

- \extracolsep This command needs to expand during the tabular preamble construction so can't be robust.

167 \def\extracolsep#1{\tabskip #1\relax}

(*End definition for \extracolsep.*)

\array(*array*)

168 \def\array{\let\@acol\@arrayacol \let\@classz\@arrayclassz  
169 \let\@classiv\@arrayclassiv  
170 \let\\@\@arraycr\let\@haligno\@empty\@tabarray}

(*End definition for \array.*)

\endarray

\endtabular 171 \def\endarray{\crcr\egroup\egroup}  
\endtabular\* 172 \def\endtabular{\crcr\egroup\egroup \$}\egroup  
173 \expandafter \let \csname endtabular\*\endcsname = \endtabular

(*End definition for \endarray, \endtabular, and \endtabular\*.*)

\tabu\tabular(*tabular*)

174 \def\tabular{\let\@haligno\@empty\@tabular}

(*End definition for \tabular.*)

\tabular\*

- Note that the change to use \setlength slightly alters the timing of the expansion and use of the length in #1 but this is very unlikely to have any practical effect.

175 \namedef{tabular\*}{#1}{%  
176 \setlength\dimen@{#1}{%  
177 \edef\@haligno{to\the\dimen@}\@tabular}

(*End definition for \tabular\*.*)

\@tabular

178 \def\@tabular{\leavevmode \hbox \bgroup \$\let\@acol\@tabacol  
179 \let\@classz\@tabclassz  
180 \let\@classiv\@tabclassiv \let\\@\@tabularcr\@tabarray}

(*End definition for \@tabular.*)

\@tabarray RmS 91/11/04 added \m@th.

181 \def\@tabarray{\m@th\@ifnextchar[\@array{\@array[c]}}

(*End definition for \@tabarray.*)

RmS 1993/11/03 changed \halign to \ialign and removed superfluous \tabskip assignment

\@array

182 \def\@array[#1]{#2}{%  
183 \if #1t\vtop \else \if#1b\vbox \else \vcenter \fi\fi

```
184 \bgroup
```

This next bit of code sets up the strut and then builds the `\halign` and its preamble according to the specification in the second argument.

This code has been moved inside the box. A side effect of this has been to expose what was a buglet in the previous version: since the `\@arstrut` below is expanded and contains an `\ifmmode` then it could produce an unnecessary extra box in every row, thus wasting ‘lots of’ main memory.

```
185 \setbox\@arstrutbox\hbox{%
186 \vrule \height\arraystretch\ht\strutbox
187 \depth\arraystretch \dp\strutbox
188 \width\z@\%
189 \mkpream{#2}%
190 \edef\@preamble{%
191 \ialign \noexpand\@halignto
192 \bgroup \@arstrut \@preamble \tabskip\z@skip \cr}%
193 }
```

That is the end of setting up the preamble; now we reset things before executing the `\halign` built-up in `\@preamble`. The restorations could be done by introducing an extra group, thus saving tokens.

```
193 \let\@startpbox\@startpbox \let\@endpbox\@endpbox
194 \let\tabularnewline\\%
195 \let\par\empty
196 \let\sharp##%
197 \set@typeset@protect
198 \lineskip\z@skip\baselineskip\z@skip
```

If the parsing of the preamble goes wrong there may be some characters left which TeX then tries to typeset, i.e., we would be in horizontal mode. That would produce an endless loop because the `\halign` expects vertical mode thus issues a `\par` but that is a no-op at this point. So we better test this case issue some error message and make a crude recovery by ending that horizontal mode with force. A better fix would be to ensure that we never pick up more than a single character token (not done).

```
199 \ifhmode \preamerr\z@ \par\fi
200 \preamble}
```

(End definition for `\@array`.)

`\@arraycr` Array version of `\``.

```
201 \protected\def\@arraycr{%
202 ${\ifnum0='}\fi\@ifstar\@xarraycr\@xarraycr}
```

(End definition for `\@arraycr`.)

`\@arraycr`

```
203 \def\@xarraycr{\@ifnextchar[\@garraycr{\ifnum0='{\fi}{$}\cr}}
```

(End definition for `\@arraycr`.)

`\@garraycr`

```
204 \def\@garraycr[#1]{%
205 \ifnum0='{\fi}{$}\ifdim #1>\z@ \@xarraycr[#1]\else
206 \@yarraycr[#1]\fi}
```

(End definition for `\@garraycr`.)

```

\tabularnewline Tabular version of \\.
207 \let\tabularnewline\relax
(End definition for \tabularnewline.)

\@tabularcr
208 \protected\def\@tabularcr{%
209 {\ifnum0='}\fi\@ifstar\@xtabularcr\@xtabularcr}
(End definition for \@tabularcr.)

\@xtabularcr
210 \def\@xtabularcr{\@ifnextchar[\@argtabularcr{\ifnum0='{\fi}\cr}}
(End definition for \@xtabularcr.)

\@argtabularcr
211 \def\@argtabularcr[#1]{%
212 \ifnum0='{\fi}%
213 \ifdim #1>\z@%
214 \unskip\@xargarraycr[#1]%
215 \else%
216 \@yargarraycr[#1]%
217 \fi}
(End definition for \@argtabularcr.)

\@xargarraycr
218 \def\@xargarraycr#1{\@tempdima #1\advance\@tempdima \dp \arstrutbox
219 \vrule \height\z@ \depth\@tempdima \width\z@ \cr}
(End definition for \@xargarraycr.)

\@yargarraycr
220 </2ekernel>
221 <*2ekernel | latexrelease>
222 <latexrelease>\IncludeInRelease{2020/10/01}%
223 <latexrelease> {\@yargarraycr}{tabular support calc syntax}%
224 \def\@yargarraycr#1{\cr\noalign{\vspace@calcify{#1}}}
225 </2ekernel | latexrelease>
226 <latexrelease>\EndIncludeInRelease
227 <latexrelease>\IncludeInRelease{0000/00/00}%
228 <latexrelease> {\@yargarraycr}{tabular support calc syntax}%
229 <latexrelease>
230 <latexrelease>\def\@yargarraycr#1{\cr\noalign{\vskip #1}}
231 <latexrelease>\EndIncludeInRelease
232 <*2ekernel>
(End definition for \@yargarraycr.)

```

\multicolumn *Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*

```
\multicolumn{NUMBER}{FORMAT}{ITEM} ==
BEGIN
\multispan{NUMBER}
\begingroup
\caddamp == null
\mkpream{FORMAT}
\sharp == ITEM
\protect == \relax
\startpbox == \@@startpbox
\endpbox == \@@endpbox
\carstrut
\preamble
\endgroup
END
```

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

The command \def\caddamp{} was removed from \multicolumn on 6 Dec 86 because it caused embedded array environments not to work. I think that it was included originally to prevent an error message if the 2nd argument to the \multicolumn command had two column specifiers.

8 Feb 89 — \hbox{} added after \preamble to correct bug that occurred if \multicolumn preceded \\[D] with D > 0, caused by \\[] command doing an \unskip, which removed \tabcolsep glue inserted by \multicolumn.

This has been made long so that, for example, a p-column can contain multiple paragraphs; maybe the arguments of @-expressions should also be able to contain multiple paragraphs.

```
233 \long\def\multicolumn#1#2#3{\multispan{#1}\begingroup
234 \mkpream{#2}%
235 \def\sharp{#3}\set@typeset@protect
236 \let\startpbox\@@startpbox\let\endpbox\@@endpbox
237 \carstrut \preamble\hbox{}\endgroup\ignorespaces}
```

(End definition for \multicolumn.)

*Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*

Codes for classes and character numbers of array, tabular and multicolumn arguments.

| Character | Class | Number |
|-----------|-------|--------|
| c         | 0     | 0      |
| l         | 0     | 1      |
| r         | 0     | 2      |
|           | 1     | -      |
| @         | 2     | -      |
| p         | 3     | -      |
| {@-exp}   | 4     | -      |
| {p-arg}   | 5     | -      |

\@testpach \foo : expands \foo, which should be an array parameter

token, and sets \chclass and \chnum to its class and number. Uses \lastchclass to distinguish 4 and 5

Preamble error codes

- 0: 'illegal character'
- 1: 'Missing @-exp'
- 2: 'Missing p-arg'

```
\@addamp ==
BEGIN if @firststamp = true then @firststamp := false
else & fi
END

\@mkpream TOKENLIST ==
BEGIN
 @firststamp := T
 \lastchclass := 6
 \@preamble == null
 \@sharp == \relax
 \@protect == BEGIN \noexpand\protect\noexpand END
 \@startpbox == \relax
 \@endpbox == \relax
 \@expast{TOKENLIST}
 for \@nextchar := expand(\reserved@a)
 do \@testpach{\@nextchar}
 case of \chclass
 0 -> \@classz
 1 -> \@classi
 ...
 5 -> \@classv
 end case
 \lastchclass := \chclass
 od
 case of \lastchclass
 0 -> \hskip \arraycolsep % lrc
 1 -> % |
 2 -> \@preamerr1 % 'Missing @-exp' % @
 3 -> \@preamerr2 % 'Missing p-arg' % p
 4 -> % @-exp
 5 -> \hskip \arraycolsep % p-exp
 end case
END

\@arrayclassz ==
BEGIN
 \@preamble := \@preamble *
 case of \lastchclass
 0 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
 1 -> \@addamp \hskip \arraycolsep
 2 -> % impossible
```

```

 3 -> % impossible
 4 -> \@addamp
 5 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
 6 -> \@addamp \hskip \arraycolsep
 end case
 * case of \@chnum
 0 -> \hfil$\relax\sharp$\hfil
 1 -> $\relax\sharp$\hfil
 2 -> \hfil$\relax\sharp$\hfil
 end case
END

\@tabclassz == similar to \@arrayclassz

\@classi ==
BEGIN
 \@preamble := \@preamble *
 case of \@lastchclass
 0 -> \hskip \arraycolsep \arrayrule
 1 -> \hskip \doublerulesep \arrayrule
 2 -> % impossible
 3 -> % impossible
 4 -> \arrayrule
 5 -> \hskip \arraycolsep \arrayrule
 6 -> \@arrayrule
 end case
END

\@classii ==
BEGIN
 \@preamble := \@preamble *
 case of \@lastchclass
 0 ->
 1 -> \hskip .5\arrayrulewidth
 2 -> % impossible
 else ->
 end case
END

\@classiii ==
BEGIN
 \@preamble := \@preamble *
 case of \@lastchclass
 0 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
 1 -> \@addamp \hskip \arraycolsep
 2 -> % impossible
 3 -> % impossible
 4 -> \@addamp
 5 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
 6 -> \@addamp \hskip \arraycolsep

```

```

 end case
END

\@arrayclassiv ==
BEGIN \@preamble := \@preamble * $ \@nextchar$ END

\@tabclassiv == same as \@arrayclassv except without the $... $

\@classv ==
BEGIN
 \@preamble :=
 \@preamble * \@startpbox{\@nextchar}\ignorespaces\@sharp
 \@endpbox
END

\@expast{S}:
Sets \reserved@a := S with all instances of *{N}{STRING}
replaced by N copies of STRING, where N > 0. An *
appearing inside braces is ignored, but *-expressions
inside STRING are expanded, so nested *-expressions are
handled properly.

\@expast{S} == BEGIN \@xexpast S *0x\@c END

\@xexpast S1 *{N}{S2} S3 \@c ==
BEGIN
 \reserved@a := S1
 \tempcnta := N
 if \tempcnta > 0
 then while \tempcnta > 0 do \reserved@a := \reserved@a S2
 \tempcnta := \tempcnta - 1 od
 \reserved@b == \@xexpast
 else \reserved@b == \@xexnoop
 fi
 \expandafter \reserved@b \reserved@a S3 \@c
END
End of historical LATEX 2.09 comments.
```

```
\@xexnoop
238 \def\@xexnoop #1\@c{}

(End definition for \@xexnoop.)

\@expast
239 \def\@expast#1{\@xexpast #1*0x\@c}

(End definition for \@expast.)
```

```

\@xexpast

240 \def\@xexpast#1##2##3##4\@@{%
241 \edef\reserved@a{\#1}%
242 \tempcnta#2\relax
243 \ifnum\tempcnta>\z@%
244 \whilenum\tempcnta>\z@\do
245 {\edef\reserved@a{\reserved@a\#3}\advance\tempcnta \m@ne}%
246 \let\reserved@b\@xexpast
247 \else
248 \let\reserved@b\@x noop
249 \fi
250 \expandafter\reserved@b\reserved@a \#4\@@}

```

(End definition for `\@xexpast`.)

```

\if@firstamp
\@addamp 251 \newif\if@firstamp

252 \def\@addamp{%
253 \if@firstamp
254 \if@firstampfalse
255 \else
256 \edef\@preamble{\@preamble &}%
257 \fi}

```

(End definition for `\if@firstamp` and `\@addamp`.)

```

\@arrayacol
\@tabacol 258 \def\@arrayacol{\edef\@preamble{\@preamble \hskip \arraycolsep}}
\@ampacol 259 \def\@tabacol{\edef\@preamble{\@preamble \hskip \tabcolsep}}
\@cacolampacol 260 \def\@ampacol{\@addamp \@acol}
261 \def\@cacolampacol{\@acol\@addamp\@acol}

```

(End definition for `\@arrayacol` and others.)

```

\@mkpream
262 \def\@mkpream#1{\@firstamptrue\@lastchclass6
263 \let\@preamble\empty
264 \let\protect\unexpandable\protect
265 \let\sharp\relax
266 \let\@startpbox\relax\let\@endpbox\relax
267 \expandafter\@tfor \expandafter
268 \nextchar \expandafter:\expandafter=\reserved@a\do
269 {\@testpach\@nextchar
270 \ifcase \chclass \classz \or \classi \or \classii \or \classiii
271 \or \classiv \or \classv \fi\@lastchclass\chclass}%
272 \ifcase \lastchclass \@acol
273 \or \or \preamerr \one\or \preamerr \tw@ \or \or \@acol \fi}

```

(End definition for `\@mkpream`.)

```

\@arrayclassz

275 \def\@arrayclassz{\ifcase \lastchclass \acolampacol \or \campacol \or
276 \or \or \addamp \or
277 \acolampacol \or \firststampfalse \acol \fi
278 \edef\@preamble{\@preamble
279 \ifcase \chnum
280 \hfil$\relax\sharp$\hfil \or $\relax\sharp$\hfil
281 \or \hfil$\relax\sharp$\hfil\fi\}

```

(End definition for \@arrayclassz.)

\@tabclassz RmS 91/08/14 inserted extra braces around entry for NFSS

```

282 \def\@tabclassz{%
283 \ifcase\lastchclass
284 \acolampacol
285 \or
286 \campacol
287 \or
288 \or
289 \or
290 \addamp
291 \or
292 \acolampacol
293 \or
294 \firststampfalse\acol
295 \fi
296 \edef\@preamble{%
297 \@preamble{%
298 \ifcase\chnum
299 \hfil
300 \hskip1sp%
301 \ignorespaces\sharp\unskip\hfil
302 \or
303 \hskip1sp\ignorespaces\sharp\unskip\hfil
304 \or
305 \hfil\hskip1sp\ignorespaces\sharp\unskip
306 \fi}}\}

```

(End definition for \@tabclassz.)

```

\@classi

307 \def\@classi{%
308 \ifcase\lastchclass
309 \acol\arrayrule
310 \or
311 \addtopreamble{\hskip \doublerulesep}\arrayrule
312 \or
313 \or
314 \or
315 \arrayrule
316 \or
317 \acol\arrayrule
318 \or

```

```

319 \@arrayrule
320 \fi}

(End definition for \@classi.)
```

\@classii

```

321 \def\@classii{%
322 \ifcase\@lastchclass
323 \or
324 \addtopreamble{\hspace{.5\arrayrulewidth}%
325 \fi}

(End definition for \@classii.)
```

\@classiii

```

326 \def\@classiii{\ifcase \@lastchclass \acolampacol \or
327 \addamp\acol \or
328 \or \or \addamp \or
329 \acolampacol \or \ampacol \fi}

(End definition for \@classiii.)
```

\@tabclassiv

```

330 \def\@tabclassiv{\addtopreamble\@nextchar}

(End definition for \@tabclassiv.)
```

\@arrayclassiv

```

331 \def\@arrayclassiv{\addtopreamble{$\@nextchar$} }
```

(End definition for \@arrayclassiv.)

\@classv

```

332 \def\@classv{\addtopreamble{\startpbox{\@nextchar}\ignorespaces
333 \sharp\endpbox}}
```

(End definition for \@classv.)

\@addtopreamble

```

334 \def\@addtopreamble#1{\edef\@preamble{\@preamble #1}}
```

(End definition for \@addtopreamble.)

\@chclass

\@lastchclass

```

335 \newcount\@chclass
```

\@chnum

```

336 \newcount\@lastchclass
337 \newcount\@chnum
```

(End definition for \@chclass, \@lastchclass, and \@chnum.)

\arraycolsep

\tabcolsep

```

338 \newdimen\arraycolsep
```

\arrayrulewidth

```

339 \newdimen\tabcolsep
```

\doublerulesep

```

340 \newdimen\arrayrulewidth
341 \newdimen\doublerulesep
```

(End definition for `\arraycolsep` and others.)

`\arraystretch`

342   `\def\arraystretch{1} % Default value.`

(End definition for `\arraystretch`.)

`\@arstrutbox`

343   `\@arstrut \newbox\@arstrutbox`

344   `\def\@arstrut{%`

345    `\relax\ifmmode\copy\@arstrutbox\else\unhcopy\@arstrutbox\fi}`

(End definition for `\@arstrutbox` and `\@arstrut`.)

`\@arrayrule`

346   `\def\@arrayrule{\@addtopreamble{\hskip -.5\arrayrulewidth`

347      `\vrule \width \arrayrulewidth\hskip -.5\arrayrulewidth}}`

(End definition for `\@arrayrule`.)

`\@testpatch`

348   `\def\@testpatch#1{\@chclass \ifnum \@lastchclass=\tw@ 4 \else`

349      `\ifnum \@lastchclass=3 5 \else`

350       `\z@ \if #1c\@chnum \z@ \else`

351          `\if #11\@chnum \one \else`

352           `\if #1r\@chnum \tw@ \else`

353           `\@chclass \if #1|\one \else`

354              `\if #1@\tw@ \else`

355              `\if #1p3 \else \z@ \@preamerr 0\fi`

356          `\fi \fi \fi \fi \fi \fi`

357   `\fi}`

(End definition for `\@testpatch`.)

`\hline`

358   `\def\hline{%`

359    `\noalign{\ifnum0='}\fi\hrule \height \arrayrulewidth \futurelet`

360    `\reserved@a\@xhline}`

(End definition for `\hline`.)

`\@xhline`

361   `\def\@xhline{\ifx\reserved@a\hline`

362      `\vskip\doublerulesep`

Measure from the middle of the rules.

363      `\vskip-\arrayrulewidth`

364      `\fi`

365    `\ifnum0='{\fi}}`

(End definition for `\@xhline`.)

`\vline`

366   `\def\vline{\vrule \width \arrayrulewidth}`

(End definition for `\vline`.)

`\cline` The old L<sup>A</sup>T<sub>E</sub>X2.09 implementation of `\cline` used up quite a lot of memory and two precious count registers. This new (1995/09/14) implementation does not use any count registers. It is coded in a way that depends heavily on the definition of `\multispan` so that command has been moved here from the file `lplain.dtx`.

These counters are no longer declared.

```

\newcount\@cla
\newcount\@clb

367 \def\cline#1{\@cline#1\@nil}

368 \def\@cline#1-#2\@nil{%
369 \omit

```

Use the counter from `\multispan`.

```

370 \multicnt#1%
371 \advance\multispan\m@ne
372 \ifnum\multicnt=\@ne\@firstofone{\&\omit}\fi
373 \multicnt#2%
374 \advance\multicnt-#1%
375 \advance\multispan\@ne

```

The original had `\unskip` at this point, but how could a skip get here ???

```

376 \leaders\hrule\@height\arrayrulewidth\hfill
377 \cr

```

This is back spacing is fairly horrible, but it is what happened in the old version... An alternative would be to make `\cline` look ahead for a following `\cline` as does `\hline`. This would alter the spacing in existing documents so keep the old version in the kernel. Perhaps a package should do this differently.

```
378 \noalign{\vskip-\arrayrulewidth}}
```

(End definition for `\cline` and `\@cline`.)

`\mscount` The `\mscount` counter is no longer declared, saving a csname and a register. It is declared in compatibility mode.

(End definition for `\mscount`.)

`\multispan` Modify `\multispan` slightly from its plain T<sub>E</sub>X definition to allow more efficient code sharing with `\multicolumn`. Also share a count register with `\multiput`.

```

\sp@n 379 \def\multispan{\omit\@multispan}
380 \def\@multispan#1{%
381 \multicnt#1\relax
382 \loop\ifnum\multicnt>\@ne \sp@n\repeat}
383 \def\sp@n{\span\omit\advance\multicnt\m@ne}

```

(End definition for `\multispan`, `\@multispan`, and `\sp@n`.)

`\@startpbox` Helper macros for ‘p’ columns.

```

\@endpbox 384 \def\@startpbox#1{\vtop\bgroup \setlength\hsize{#1}\@arrayparboxrestore}
 \@startpbox{width} text \egroup is essentially \parbox{width}{text}
 \@endpbox is essentially \unskip \strut \par \egroup\hfil (Changed 14 Jan 89)
 (changed again 1994/05/13)

```

```
384 \def\@startpbox#1{\vtop\bgroup \setlength\hsize{#1}\@arrayparboxrestore}
```

```
385 \def\@endpbox{\@finalstrut\@arstrutbox\par\egroup\hfil}
14 Jan 89: Def of \@endpbox changed from
\def\@endpbox{\par\vskip\dp\@arstrutbox\egroup\hfil}
so vertical spacing works out right if the last line of a ‘p’ entry has a descender.
```

(*End definition for \@startpbox and \@endpbox.*)

```
\@@startpbox
\@@endpbox
386 \let\@@startpbox=\@startpbox
387 \let\@@endpbox=\@endpbox
```

(*End definition for \@startpbox and \@endpbox.*)

```
388 </2ekernel>
```

# File M

## ltpictur.dtx

### 1 Picture Mode

Picture mode commands. In addition to the commands available in L<sup>A</sup>T<sub>E</sub>X2.09, This section adds the new \qbezier command for drawing curves.

\qbezier \qbezier[ $\langle N \rangle$ ] ( $\langle AX,AY \rangle$ ) ( $\langle BX,BY \rangle$ ) ( $\langle CX,CY \rangle$ ) plots a quadratic Bezier curve from ( $\langle AX,AY \rangle$ ) to ( $\langle CX,CY \rangle$ ), with ( $\langle BX,BY \rangle$ ) as the third Bezier point, using  $N+1$  points equally spaced parametrically. If  $N = 0$  (the default value), then a sufficient number of points are used to draw a connected curve—except that at most \qbeziermax+1 points are drawn. A “point” is a square of side \@wholewidth.

\bezier In addition, to be compatible with the old **bezier** package, a variant of this command, \bezier, is defined, in which the first argument is not optional.

*Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*

|              |                               |
|--------------|-------------------------------|
| \unitlength  | = value of dimension argument |
| \@wholewidth | = current line width          |
| \@halfwidth  | = half of current line width  |
| \@linefnt    | = font for drawing lines      |
| \@circlefnt  | = font for drawing circles    |

\linethickness{DIM} : Sets the width of horizontal and vertical lines in a picture to DIM. Does not change width of slanted lines or circles. Width of all lines reset by \thinlines and \thicklines

```
\picture(XSIZE,YSIZE)(XORG,YORG)
BEGIN
 \@picht := L YSIZE * \unitlength
 box \@picbox :=
 \hb@xt@ XSIZE * \unitlength
 {\hskip -XORG * \unitlength
 \lower YORG * \unitlength
 \hbox{
 \ignorespaces %% added 13 June 89
 }
 END

\endpicture ==
BEGIN
 } \hss }
height of \@picbox := \@picht
depth of \@picbox := 0
\mbox{\box\@picbox} %% change 26 Aug 91
END

\put(X, Y){OBJ} ==
BEGIN
```

```

\@killglue
\raise Y * \unitlength \hb@xt@ 0pt { \hskip X * \unitlength
 OBJ \hss }
\ignorespaces
END

\multiput(X,Y)(DELX,DELY){N}{OBJ} ==
BEGIN
\@killglue
\@multicnt := N
\@xdim := X * \unitlength
\@ydim := Y * \unitlength
while \@multicnt > 0
 do \raise \@ydim \hb@xt@ 0pt { \hskip \@xdim
 OBJ \hss }
\@multicnt := \@multicnt - 1
\@xdim := \@xdim + DELX * \unitlength
\@ydim := \@ydim + DELY * \unitlength
od
\ignorespaces
END

\shortstack[POS]{TEXT} : Makes a \vbox containing TEXT stacked as
a one-column array, positioned l, r or c as indicated by POS.

```

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

The ‘2ekernel’ code ensures that a \usepackage{autopict} is essentially ignored if a ‘full’ format is being used that has picture mode already in the format.

1 <2ekernel>\expandafter\let\csname ver@autopict.sty\endcsname\fmtversion

```

\@wholewidth
\@halfwidth
 2 <*2ekernel>
 3 \newdimen\@wholewidth
 4 \newdimen\@halfwidth

```

*(End definition for \@wholewidth and \@halfwidth.)*

```

\unitlength
 5 \newdimen\unitlength \unitlength =1pt

```

*(End definition for \unitlength.)*

```

\@picbox
\@picht
 6 \newbox\@picbox
 7 \newdimen\@picht

```

*(End definition for \@picbox and \@picht.)*

\@defaultunitsset Set a length register, #1, accepting number or an etex length expression, #2, with default unit, #3.

The register name in #1 can be prefixed by \advance so that the register is incremented by the supplied value.

```
\@defaultunitsset{\advance{@vxx}{\textwidth-15pt}\unitlength
#3 can be a literal unit such as cm or a length register such as \unitlength.
```

This is used in all `picture` commands that take picture coordinates. So `\put(2,2)` as previously but now `\put(\textwidth-5cm,0.4\textheight)` Note that you can only use expressions with lengths, `\put(1+2,0)` is not supported.

```
8 </2ekernel>
9 <*2ekernel | latexrelease>
10 <latexrelease>\IncludeInRelease{2020/10/01}%
11 <latexrelease> {\@defaultunitsset{default units}%
12 \def\@defaultunitsset#1#2#3{%
13 \@defaultunits#1\dimexpr#2#3\relax\relax\@nnil}
14 </2ekernel | latexrelease>

15 <latexrelease>\EndIncludeInRelease
16 <latexrelease>\IncludeInRelease{0000/00/00}%
17 <latexrelease> {\@defaultunitsset{default units}%
18 <latexrelease>\let\@defaultunitsset\@undefined
19 <latexrelease>\EndIncludeInRelease
20 <*2ekernel>
```

(End definition for `\@defaultunitsset`.)

`pict\picture`) #1 should be white space.

#1 should be a ( (eating any white space before the bracket),

```
\pictur@ 21 \long\def\picture#1{\pictur@#1}
22 \def\pictur@(#1){%
23 \@ifnextchar({\@picture(#1)}{\@picture(#1)(0,0)})
```

(End definition for `\picture` and `\pictur@`.)

\@picture

```
24 </2ekernel>
25 <*2ekernel | latexrelease>
26 <latexrelease>\IncludeInRelease{2020/10/01}%
27 <latexrelease> {\@picture{default units}%
28 \def\@picture(#1,#2)(#3,#4){%
29 \@defaultunitsset\@picht{#2}\unitlength
30 \@defaultunitsset\@tempdimc{#1}\unitlength
31 \setbox\@picbox\hb@xt@\@tempdimc\bgroun
32 \@defaultunitsset\@tempdimc{#3}\unitlength
33 \hskip -\@tempdimc
34 \@defaultunitsset\@tempdimc{#4}\unitlength
35 \lower\@tempdimc\hbox\bgroun
36 \ignorespaces}
37 </2ekernel | latexrelease>
```

```

38 \end{macro}
39 \end{macro}
40 \def\@picture#1#2#3#4{%
41 \begingroup\hss\egroup
42 \setbox\@picbox\hb@xt@#1\unitlength\bgroup
43 \hspace{-#3\unitlength}
44 \lower #4\unitlength\hbox\bgroup
45 \ignorespaces}
46 \end{macro}
47 \end{macro}
48 \end{macro}

(End definition for \@picture.)
```

\endpicture

```

49 \def\endpicture{%
50 \begingroup\hss\egroup
51 \ht\@picbox\@picht\dp\@picbox\z@%
52 \mbox{\box\@picbox}}
```

(End definition for \endpicture.)

In the definitions of \put and \multiput, \hspace was replaced by \kern just in case arg #3 = “plus”. (Bug detected by Don Knuth. changed 20 Jul 87).

```

53 \end{macro}
54 \end{macro}
55 \end{macro}
56 \def\put#1#2#3{%
57 \expandafter\let\csname put \endcsname\@undefined
58 \long\def\put#1#2#3{%
59 \killglue
60 \defaultunitsset\tempdimc{#2}\unitlength
61 \raise\tempdimc
62 \hb@xt@\z@{%
63 \defaultunitsset\tempdimc{#1}\unitlength
64 \kern\tempdimc
65 #3\hss}%
66 \ignorespaces}
67 \end{macro}
68 \end{macro}
69 \end{macro}
70 \def\put#1#2#3{%
71 \expandafter\let\csname put \endcsname\@undefined
72 \long\def\put#1#2#3{%
73 \killglue\raise#2\unitlength
74 \hb@xt@\z@{\kern#1\unitlength #3\hss}%
75 \ignorespaces}
76 \end{macro}
77 \end{macro}

\multiput #3 had better be a .
```

```

78 \end{macro}
79 \end{macro}
80 \end{macro}
```

```

81 \begin{macro}{\multiput}
82 \expandafter\let\csname multiput \endcsname\undefined
83 \def\multiput(#1,#2){%
84 \ifdim#1=0pt \def\unitlength{\unitlength}%
85 \ifdim#2=0pt \def\unitlength{\unitlength}%
86 \multiput{}%
87 }%
```

(*End definition for \multiput.*)

\@multiput

```

98 \begin{macro}{\@multiput}
99 \expandafter\let\csname @multiput \endcsname\undefined
100 \def\@multiput(#1,#2){%
101 \ifdim#1=0pt \def\unitlength{\unitlength}%
102 \ifdim#2=0pt \def\unitlength{\unitlength}%
103 \killglue\multicnt #3\relax
104 \whilenum \multicnt >\z@\do
105 {\raise\ydim\hb@xt@.0{\kern\unitlength}%
106 \advance\multicnt\m@ne
107 \ifdim#2>0pt \advance\ydim\unitlength%
108 \else \advance\ydim-\unitlength%
109 \fi
110 }%
```

(*End definition for \@multiput.*)

\@killglue

```

123 \def\@killglue{\unskip\whiledim \lastskip >\z@\do{\unskip}}
```

(*End definition for \@killglue.*)

```

\thinlines
\thicklines 124 \DeclareRobustCommand\thinlines{\let\@linefnt\tenln
125 \let\@circlefnt\tencirc
126 \@wholewidth\fontdimen8\tenln \@halfwidth .5\@wholewidth}
127 \DeclareRobustCommand\thicklines{\let\@linefnt\tenlnw
128 \let\@circlefnt\tencircw
129 \@wholewidth\fontdimen8\tenlnw \@halfwidth .5\@wholewidth}

(End definition for \thinlines and \thicklines.)

\linethickness
130 \DeclareRobustCommand*\linethickness[1]
131 {\@wholewidth #1\relax \@halfwidth .5\@wholewidth \ignorespaces}

(End definition for \linethickness.)

\isshortstack
132 \def\shortstack{\@ifnextchar[\@shortstack{\@shortstack[c]}}
(End definition for \isshortstack.)

\@ishortstack
133 \def\@shortstack[#1]{%
134 \leavevmode
135 \vbox\bgroun
136 \baselineskip-\p@\lineskip 3\p@
137 \let\mb@l\hss\let\mb@r\hss
138 \expandafter\let\csname mb@#1\endcsname\relax
139 \let\\@stackcr
140 \@ishortstack}

(End definition for \@ishortstack.)

\@ishortstack
141 \def\@ishortstack#1{\ialign{\mb@l {##}\unskip\mb@r\cr #1\crcr}\egroup}
(End definition for \@ishortstack.)

\@stackcr
\@ixstackcr 142 \protected\def\@stackcr{\@ifstar\@ixstackcr\@ixstackcr}
143 \def\@ixstackcr{\@ifnextchar[\@istackcr{\cr\ignorespaces}{}}

(End definition for \@stackcr and \@ixstackcr.)

\@istackcr
144 </2ekernel>
145 <*2ekernel | latexrelease>
146 <latexrelease>\IncludeInRelease{2020/10/01}%
147 <latexrelease> {\@istackcr}{\shortstack calc support}%
148 \def\@istackcr[#1]{\cr\noalign{\@vspace@calcify{#1}}\ignorespaces}
149 </2ekernel | latexrelease>

```

```

150 \end{macro}
151 \end{macro}
152 \end{macro}
153 \end{macro}
154 \def\@istackcr[#1]{\cr\noalign{\vskip #1}\ignorespaces}
155 \end{macro}
156 {*2ekernel}

(End definition for \@istackcr.)
Historical LATEX 2.09 comments (not necessarily accurate any more):
\line(X,Y){LEN} ==
BEGIN
 \carg := X
 \yarg := Y
 \clinenlen := LEN * \unitlength
 if \carg = 0
 then \vline
 else if \yarg = 0
 then \hline
 else \sline
 if
 if
END

\sline ==
BEGIN
 if \carg < 0
 then @negarg := T
 \carg := -\carg
 \yyarg := -\yarg
 else @negarg := F
 \yyarg := \yarg
 fi
 \tempcnta := |\yyarg|
 if \tempcnta > 6
 then error: 'LATEX ERROR: Illegal \line or \vector argument.'
 \tempcnta := 0
 fi
 \box\clinechar := \hbox{\clinefnt \getlinechar(\carg,\yyarg) }
 if \yarg > 0 then \upordown = \raise
 \clnht := 0
 else \upordown = \lower
 \clnht := height of \box\clinechar
 fi
 \clnwd := width of \box\clinechar
 if @negarg
 then \hskip - width of \box\clinechar
 \reserved@a == \hskip - 2* width of box \clinechar
 else \reserved@a == \relax
 fi
% Put out integral number of line segments

```

```

while \@clnwd < \@linelen
 do \upordown \@clnht \copy\@linechar
 \reserved@a
 \@clnht := \@clnht + ht of \box\@linechar
 \@clnwd := \@clnwd + width of \box\@linechar
 od

%% Put out last segment
\@clnht := \@clnht - height of \box\@linechar
\@clnwd := \@clnwd - width of \box\@linechar
\@tempdima := \@linelen - \@clnwd
\@tempdimb := \@tempdima - width of \box\@linechar
if @negarg then \hskip -\@tempdimb
 else \hskip \@tempdimb
fi
\@tempdima := 1000 * \@tempdima
\@tempcpta := \@tempdima / width of \box\@linechar
\@tempdima := (\@tempcpta * ht of \box\@linechar)/1000
\@clnht := \@clnht + \@tempdima
if \@linelen < width of box\@linechar
 then \hskip width of box\@linechar
 else \hbox{\upordown \@clnht \copy\@linechar}
fi
END

\@hline ==
BEGIN
if \xarg < 0 then \hskip -\@linelen \fi
\vrule height \halfwidth depth \halfwidth width \@linelen
if \xarg < 0 then \hskip -\@linelen \fi
END

\@vline == if \yarg < 0 \downline else \upline fi

\@getlinechar(X,Y) ==
BEGIN
\@tempcpta := 8*X - 9
if Y > 0
 then \@tempcpta := \@tempcpta + Y
 else \@tempcpta := \@tempcpta - Y + 64
fi
\char\@tempcpta
END

\vector(X,Y){LEN} ==
BEGIN
\xarg := X
\yarg := Y
\@linelen := LEN * \unitlength

```

```

if \@xarg = 0
 then \@vvector
else if \@yarg = 0
 then \@hvector
 else \@svector
 if
 if
END

\@hvector ==
BEGIN
 \@hline
 {\@clinefnt if \@xarg < 0 then \@getlarrow(1,0)
 else \@getrarrow(1,0)
 fi}
END

\@vvector == if \@yarg < 0 \@downvector else \@upvector fi

\@svector ==
BEGIN
 \@sline
 \tempcnta := |\@yarg|
 if \tempcnta < 5
 then \hskip - width of \box\@linechar
 \@upordown \clnht \hbox
 {\@clinefnt
 if @negarg then \@getlarrow(\@xarg,\@yyarg)
 else \@getrarrow(\@xarg,\@yyarg)
 fi }
 else error: 'LATEX ERROR: Illegal \line or \vector argument.'
 fi
END

\@getlarrow(X,Y) ==
BEGIN
 if Y = 0
 then \tempcnta := '33
 else \tempcnta := 16 * X - 9
 \tempcntb := 2 * Y
 if \tempcntb > 0
 then \tempcnta := \tempcnta + \tempcntb
 else \tempcnta := \tempcnta - \tempcntb + 64
 fi
 fi
 \char\tempcnta
END

\@getrarrow(X,Y) ==
BEGIN

```

```

\@tempcntb := |Y|
case of \@tempcntb
 0 : \@tempcnta := '55
 1 : if X < 3
 then \@tempcnta := 24*X - 6
 else if X = 3
 then \@tempcnta := 49
 else \@tempcnta := 58 fi
 fi
 2 : if X < 3
 then \@tempcnta := 24*X - 3
 else \@tempcnta := 51 % X must = 3
 fi
 3 : \@tempcnta := 16*X - 2
 4 : \@tempcnta := 16*X + 7
endcase
if Y < 0
 then \@tempcnta := \@tempcnta + 64
fi
\char\@tempcnta
END

```

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

### \if@negarg

157 \newif\if@negarg

(*End definition for \if@negarg.*)

### \line

```

158 〈/2ekernel〉
159 〈*2ekernel | latexrelease〉
160 〈| latexrelease〉\IncludeInRelease{2020/10/01}%
161 〈| latexrelease〉 {\line}{default units}%
162 〈| latexrelease〉\expandafter\let\csname line \endcsname\@undefined
163 \def\line(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
164 \@defaultunitsset\@linelen{#3}\unitlength
165 \ifdim\@linelen<\z@\@badlineararg\else
166 \ifnum\@xarg =\z@ \@vline
167 \else \ifnum\@yarg =\z@ \@hline \else \@sline\fi
168 \fi
169 \fi}
170 〈/2ekernel | latexrelease〉

171 〈| latexrelease〉\EndIncludeInRelease
172 〈| latexrelease〉\IncludeInRelease{0000/00/00}%
173 〈| latexrelease〉 {\line}{default units}%
174 〈| latexrelease〉\expandafter\let\csname line \endcsname\@undefined
175 〈| latexrelease〉\def\line(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
176 〈| latexrelease〉 \@linelen #3\unitlength
177 〈| latexrelease〉 \ifdim\@linelen<\z@\@badlineararg\else
178 〈| latexrelease〉 \ifnum\@xarg =\z@ \@vline
179 〈| latexrelease〉 \else \ifnum\@yarg =\z@ \@hline \else \@sline\fi
180 〈| latexrelease〉 \fi

```

```

181 〈latexrelease〉 \fi}
182 〈latexrelease〉\EndIncludeInRelease
183 〈*2ekernel〉

```

(End definition for \line.)

\@sline

```

184 \def\@sline{%
185 \ifnum\@xarg<\z@ \negargtrue \xarg -\xarg \yyarg -\yarg
186 \else \negargfalse \yyarg \yarg \fi
187 \ifnum \yyarg >\z@ \tempcpta\yyarg \else \tempcpta -\yyarg \fi
188 \ifnum\tempcpta>6 \badlinearg\tempcpta\z@ \fi
189 \ifnum\@xarg>6 \badlinearg\@xarg \ne \fi
190 \setbox\@linechar\hbox{\@linefnt\getlinechar(\@xarg,\@yyarg)}%

```

If we have something like \line(5,5){30} the \@linechar will not contain a char and later on we will end in an infinite loop. So we check the width of the box and put in something as an emergency fix if necessary.

```

191 \ifdim\wd\@linechar=\z@
192 \setbox\@linechar\hbox{.}%
193 \badlinearg
194 \fi
195 \ifnum \yarg >\z@ \let\upordown\raise \clnht\z@
196 \else\let\upordown\lower \clnht \ht\@linechar\fi
197 \clnwd \wd\@linechar
198 \if\negarg
199 \hskip -\wd\@linechar \def\reserved@a{\hskip -2\wd\@linechar}%
200 \else
201 \let\reserved@a\relax
202 \fi
203 \whiledim \clnwd <\linelen \do
204 {\upordown\clnht\copy\@linechar
205 \reserved@a
206 \advance\clnht \ht\@linechar
207 \advance\clnwd \wd\@linechar}%
208 \advance\clnht -\ht\@linechar
209 \advance\clnwd -\wd\@linechar
210 \tempdima\linelen\advance\tempdima -\clnwd
211 \tempdimb\tempdima\advance\tempdimb -\wd\@linechar
212 \if\negarg \hskip -\tempdimb \else \hskip \tempdimb \fi
213 \multiply\tempdima \m
214 \tempcpta\tempdima
215 \tempdima \wd\@linechar \divide\tempcpta \tempdima
216 \tempdima \ht\@linechar \multiply\tempdima \tempcpta
217 \divide\tempdima \m
218 \advance\clnht \tempdima
219 \ifdim \linelen <\wd\@linechar
220 \hskip \wd\@linechar

```

Warn if line gets so short that it can't be printed. But don't warn if it is exactly zero since that was probably deliberate (e.g., to get a vector head only).

```

221 \ifdim \linelen = \z@
222 \else
223 \picture@warn
224 \fi

```

```

225 \else\@upordown\@clnht\copy\@linechar\fi}

(End definition for \@sline.)
```

```

\@hline

226 \def\@hline{\ifnum \carg <\z@ \hskip -\@linelen \fi
227 \vrule \height \halfwidth \depth \halfwidth \width \@linelen
228 \ifnum \carg <\z@ \hskip -\@linelen \fi}
```

(End definition for \@hline.)

```

\@getlinechar

229 \def\@getlinechar(#1,#2){\tempcnta#1\relax\multiply\tempcnta 8%
230 \advance\tempcnta -9\ifnum #2>\z@ \advance\tempcnta #2\relax\else
231 \advance\tempcnta -#2\relax\advance\tempcnta 64 \fi
232 \char\tempcnta}
```

(End definition for \@getlinechar.)

```

\vector

233 </2ekernel>
234 (*2ekernel | latexrelease)
235 <latexrelease>\IncludeInRelease{2020/10/01}%
236 <latexrelease> {\vector}{default units}%
237 <latexrelease>\expandafter\let\csname vector \endcsname\undefined
238 \def\vector(#1,#2){\carg #1\relax \carg #2\relax
239 \tempcnta \ifnum\carg<\z@ -\carg\else\carg\fi
240 \ifnum\tempcnta<5\relax
241 \defaultunitsset\@linelen{#3}\unitlength
242 \ifdim\@linelen<\z@\badlinearg\else
243 \ifnum\carg =\z@ \vvector
244 \else \ifnum\carg =\z@ \hvector \else \svector\fi
245 \fi
246 \fi
247 \else\badlinearg\fi}
248 </2ekernel | latexrelease>

249 <latexrelease>\EndIncludeInRelease
250 <latexrelease>\IncludeInRelease{0000/00/00}%
251 <latexrelease> {\vector}{default units}%
252 <latexrelease>\expandafter\let\csname vector \endcsname\undefined
253 <latexrelease>\def\vector(#1,#2){\carg #1\relax \carg #2\relax
254 \tempcnta \ifnum\carg<\z@ -\carg\else\carg\fi
255 \ifnum\tempcnta<5\relax
256 \@linelen #3\unitlength
257 \ifdim\@linelen<\z@\badlinearg\else
258 \ifnum\carg =\z@ \vvector
259 \else \ifnum\carg =\z@ \hvector \else \svector\fi
260 \fi
261 \fi
262 \else\badlinearg\fi}
263 <latexrelease>\EndIncludeInRelease
264 (*2ekernel)
```

(End definition for \vector.)

```

\@hvector
265 \def\@hvector{\@hline\hb@xt@\z@{\@linefnt
266 \ifnum \carg <\z@ \getlarrow(1,0)\hss\else
267 \hss\getrarrow(1,0)\fi}}
(End definition for \@hvector.)

\@vvector
268 \def\@vvector{\ifnum \carg <\z@ \downvector \else \upvector \fi}
(End definition for \@vvector.)

\@svector
269 \def\@svector{\@sline
270 \tempcnta\carg \ifnum\tempcnta <\z@ \tempcnta -\tempcnta\fi
271 \ifnum\tempcnta <5%
272 \hskip -\wd\@linechar
273 \upordown\clnht \hbox{\@linefnt \if@negarg
274 \getlarrow(\carg,\yyarg)\else \getrarrow(\carg,\yyarg)\fi}%
275 \else\badlinearg\fi}
(End definition for \@svector.)

\@getlarrow
276 \def\@getlarrow(#1,#2){\ifnum #2=\z@ \tempcnta 27 \% '33
277 \else
278 \tempcnta #1\relax\multiply\tempcnta \sixt@n
279 \advance\tempcnta -9 \tempcntb #2\relax\multiply\tempcntb \tw@%
280 \ifnum \tempcntb >\z@ \advance\tempcnta \tempcntb
281 \else\advance\tempcnta -\tempcntb\advance\tempcnta 64
282 \fi\fi\char\tempcnta}
(End definition for \@getlarrow.)

\@getrarrow
283 \def\@getrarrow(#1,#2){\tempcntb #2\relax
284 \ifnum\tempcntb <\z@ \tempcntb -\tempcntb\relax\fi
285 \ifcase \tempcntb\relax \tempcnta 45 \% '55
286 \or
287 \ifnum #1<\thr@ \tempcnta #1\relax\multiply\tempcnta
288 24\advance\tempcnta -6 \else \ifnum #1=\thr@ \tempcnta 49
289 \else\tempcnta 58 \fi\fi\or
290 \ifnum #1<\thr@ \tempcnta=#1\relax\multiply\tempcnta
291 24\advance\tempcnta -\thr@ \else \tempcnta 51 \fi\or
292 \tempcnta #1\relax\multiply\tempcnta
293 \sixt@n \advance\tempcnta -\tw@ \else
294 \tempcnta #1\relax\multiply\tempcnta
295 \sixt@n \advance\tempcnta 7 \fi\ifnum #2<\z@ \advance\tempcnta 64 \fi
296 \char\tempcnta}
(End definition for \@getrarrow.)

\@vline
297 \def\@vline{\ifnum \carg <\z@ \downline \else \upline\fi}

```

(End definition for \cvline.)

\@upline

```
298 \def\@upline{%
299 \hb@xt@z@{\hskip -\@halfwidth \vrule \@width \@wholewidth
300 \@height \@linelen \@depth \z@\hss}}}
```

(End definition for \@upline.)

\@downline

```
301 \def\@downline{%
302 \hb@xt@z@{\hskip -\@halfwidth \vrule \@width \@wholewidth
303 \@height \z@ \@depth \@linelen \hss}}
```

(End definition for \@downline.)

\@upvector

```
304 \def\@upvector{\@upline\setbox\@tempboxa\hbox{\@linefnt\char 54}%
305 \raise \@linelen \hb@xt@z@{\lower \ht\@tempboxa\box\@tempboxa\hss}}
```

(End definition for \@upvector.)

\@downvector

```
306 \def\@downvector{\@downline\lower \@linelen
307 \hb@xt@z@{\@linefnt\char 63 \% '77
308 \hss}}
```

(End definition for \@downvector.)

*Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*

```
\dashbox{D}{X,Y} ==
BEGIN
leave vertical mode
\hb@xt@ 0pt {
 \baselineskip := 0pt
 \lineskip := 0pt
%% HORIZONTAL DASHES
 \dashdim := X * \unitlength
 \dashcnt := \dashdim + 200 % to prevent roundoff error
 \dashdim := D * \unitlength
 \dashcnt := \dashcnt / \dashdim
 if \dashcnt is odd
 then \dashdim := 0pt
 \dashcnt := (\dashcnt + 1) / 2
 else \dashdim := \dashdim / 2
 \dashcnt := \dashcnt / 2 - 1
 \box\@dashbox := \hbox{\vrule height \@halfwidth
 depth \@halfwidth width \@dashdim}
 \put(0,0){\copy\@dashbox}
 \put(0,Y){\copy\@dashbox}
 \put(X,0){\hskip -\dashdim\copy\@dashbox}
 \put(X,Y){\hskip -\dashdim\box\@dashbox}
 \dashdim := 3 * \dashdim
fi
```

```

\box\@dashbox := \hbox{\vrule height \@halfwidth
 depth \@halfwidth width D * \unitlength
 \hskip D * \unitlength}

\@tempcnta := 0
\put(0,0){\hskip \@dashdim
 while \@tempcnta < \@dascnt
 do \copy\@dashbox
 \@tempcnta := \@tempcnta + 1
 od
 }
\@tempcnta := 0
\put(0,Y){\hskip \@dashdim
 while \@tempcnta < \@dascnt
 do \copy\@dashbox
 \@tempcnta := \@tempcnta + 1
 od
 }

%% vertical dashes
\@dashdim := Y * \unitlength
\@dashcnt := \@dashdim + 200 % to prevent roundoff error
\@dashdim := D * \unitlength
\@dashcnt := \@dashcnt / \@dashdim
if \@dashcnt is odd
 then \@dashdim := 0pt
 \@dashcnt := (\@dashcnt + 1) / 2
 else \@dashdim := \@dashdim / 2
 \@dashcnt := \@dashcnt / 2 - 1
\box\@dashbox := \hbox{\hskip -\@halfwidth
 \vrule width \@wholewidth
 height \@dashdim }

\put(0,0){\copy\@dashbox}
\put(X,0){\copy\@dashbox}
\put(0,Y){\lower\@dashdim\copy\@dashbox}
\put(X,Y){\lower\@dashdim\copy\@dashbox}
\@dashdim := 3 * \@dashdim
fi
\box\@dashbox := \hbox{\vrule width \@wholewidth
 height D * \unitlength } }

\@tempcnta := 0
\put(0,0){\hskip -\halfwidth
 \vbox{while \@tempcnta < \@dashcnt
 do \vskip D*\unitlength
 \copy\@dashbox
 \@tempcnta := \@tempcnta + 1
 od
 \vskip \@dashdim
 } }

\@tempcnta := 0
\put(X,0){\hskip -\halfwidth

```

```

 \vbox{while \tempcnta < \dashcnt
 do \vskip D*\unitlength
 \copy\dashbox
 \tempcnta := \tempcnta + 1
 od
 \vskip \dashdim
 }
 }
} % END DASHES

\@imakepicbox(X,Y)
END
End of historical LATEX 2.09 comments.

```

```

\Dashbox
309 {/2ekernel}
310 {*2ekernel | latexrelease}
311 {latexrelease}\IncludeInRelease{2020/10/01}%
312 {latexrelease} {\dashbox}{default units}%
313 {latexrelease}\expandafter\let\csname dashbox \endcsname\@undefined
314 \def\dashbox#1(#2,#3){\leavevmode\hb@xt@z@\baselineskip \z@skip
315 \lineskip \z@skip
316 \@defaultunitsset\@dashdim{#2}\unitlength
317 \dashcnt \@dashdim \advance\@dashcnt 200
318 \@defaultunitsset\@dashdim{#1}\unitlength
319 \divide\@dashcnt \@dashdim
320 \ifodd\@dashcnt\@dashdim \z@
321 \advance\@dashcnt \one \divide\@dashcnt \tw@
322 \else \divide\@dashdim \tw@ \divide\@dashcnt \tw@
323 \advance\@dashcnt \m@ne
324 \setbox\dashbox \hbox{\vrule \height \halfwidth \depth \halfwidth
325 \width \@dashdim}\put(0,0){\copy\dashbox}%
326 \put(0,#3){\copy\dashbox}%
327 \put(#2,0){\hskip-\@dashdim\copy\dashbox}%
328 \put(#2,#3){\hskip-\@dashdim\box\dashbox}%
329 \multiply\@dashdim \thr@@
330 \fi
331 \setbox\dashbox \hbox{%
332 \@defaultunitsset\@tempdimc{#1}\unitlength
333 \vrule \height \halfwidth \depth \halfwidth \width \@tempdimc
334 \hskip\@tempdimc}%
335 \tempcnta\z@
336 \put(0,0){\hskip\@dashdim \whilenum \tempcnta <\@dashcnt
337 \do{\copy\dashbox\advance\tempcnta \one } }\@tempcnta\z@
338 \put(0,#3){\hskip\@dashdim \whilenum \tempcnta <\@dashcnt
339 \do{\copy\dashbox\advance\tempcnta \one } }%
340 \@defaultunitsset\@dashdim{#3}\unitlength
341 \dashcnt \@dashdim \advance\@dashcnt 200
342 \@defaultunitsset\@dashdim{#1}\unitlength
343 \divide\@dashcnt \@dashdim
344 \ifodd\@dashcnt\@dashdim \z@
345 \advance\@dashcnt \one \divide\@dashcnt \tw@
346 \else

```

```

347 \divide\@dashdim \tw@ \divide\@dashcnt \tw@
348 \advance\@dashcnt \m@ne
349 \setbox\@dashbox\hbox{\hskip -\@halfwidth
350 \vrule \@width \@wholewidth
351 \@height \@dashdim}\put(0,0){\copy\@dashbox}%
352 \put(#2,0){\copy\@dashbox}%
353 \put(0,#3){\lower\@dashdim\copy\@dashbox}%
354 \put(#2,#3){\lower\@dashdim\copy\@dashbox}%
355 \multiply\@dashdim \thr@@
356 \fi
357 \@defaultunitsset\@tempdimb{#1}\unitlength
358 \setbox\@dashbox\hbox{%
359 \vrule \@width \@wholewidth \@height\@tempdimb}%
360 \z@\@tempcpta\z@
361 \put(0,0){\hskip -\@halfwidth \vbox{\@whilenum \@tempcpta <\@dashcnt
362 \do{\vskip\@tempdimb\copy\@dashbox\advance\@tempcpta \@ne }%
363 \vskip\@dashdim}}\@tempcpta\z@
364 \put(#2,0){\hskip -\@halfwidth \vbox{\@whilenum \@tempcpta<\@dashcnt
365 \do{\vskip\@tempdimb\copy\@dashbox\advance\@tempcpta \@ne }%
366 \vskip\@dashdim}}\makepicbox(#2,#3)}
367
```

`</2ekernel | latexrelease>`

`<| latexrelease>\EndIncludeInRelease`

`<| latexrelease>\IncludeInRelease{0000/00/00}%`

`<| latexrelease> \{\@dashbox\}{default units}%
370 <| latexrelease> \expandafter\let\csname dashbox \endcsname\@undefined`

`<| latexrelease> \def\dashbox#1(#2,#3){%
373 <| latexrelease> \leavevmode\hb@xt@z@{\baselineskip \z@skip
374 <| latexrelease> \lineskip \z@skip
375 <| latexrelease> \@dashdim #2\unitlength
376 <| latexrelease> \@dashcnt \@dashdim \advance\@dashcnt 200
377 <| latexrelease> \@dashdim #1\unitlength\divide\@dashcnt \@dashdim
378 <| latexrelease> \ifodd\@dashcnt\@dashdim \z@
379 <| latexrelease> \advance\@dashcnt \@ne \divide\@dashdim \tw@ \divide\@dashcnt \tw@
380 <| latexrelease> \else \divide\@dashdim \tw@ \divide\@dashcnt \tw@ \divide\@dashcnt \tw@
381 <| latexrelease> \advance\@dashcnt \m@ne
382 <| latexrelease> \setbox\@dashbox \hbox{%
383 <| latexrelease> \vrule \@height \@halfwidth \@depth \@halfwidth
384 <| latexrelease> \@width \@dashdim}\put(0,0){\copy\@dashbox}%
385 <| latexrelease> \put(0,#3){\copy\@dashbox}%
386 <| latexrelease> \put(#2,0){\hskip-\@dashdim\copy\@dashbox}%
387 <| latexrelease> \put(#2,#3){\hskip-\@dashdim\box\@dashbox}%
388 <| latexrelease> \multiply\@dashdim \thr@@
389 <| latexrelease> \fi
390 <| latexrelease> \setbox\@dashbox \hbox{%
391 <| latexrelease> \vrule \@height \@halfwidth \@depth \@halfwidth
392 <| latexrelease> \@width #1\unitlength\hskip #1\unitlength}\@tempcpta\z@
393 <| latexrelease> \put(0,0){\hskip\@dashdim \@whilenum \@tempcpta <\@dashcnt
394 <| latexrelease> \do{\copy\@dashbox\advance\@tempcpta \@ne }%\@tempcpta\z@
395 <| latexrelease> \put(0,#3){\hskip\@dashdim \@whilenum \@tempcpta <\@dashcnt
396 <| latexrelease> \do{\copy\@dashbox\advance\@tempcpta \@ne }%}
397 <| latexrelease> \@dashdim #3\unitlength
398 <| latexrelease> \@dashcnt \@dashdim \advance\@dashcnt 200
399 <| latexrelease> \@dashdim #1\unitlength\divide\@dashcnt \@dashdim
400 <| latexrelease> \ifodd\@dashcnt \@dashdim \z@`

```

401 〈latexrelease〉\advance\@dashcnt \@ne \divide\@dashcnt \tw@
402 〈latexrelease〉\else
403 〈latexrelease〉\divide\@dashdim \tw@ \divide\@dashcnt \tw@
404 〈latexrelease〉\advance\@dashcnt \m@ne
405 〈latexrelease〉\setbox\@dashbox\hbox{\hskip -\@halfwidth
406 〈latexrelease〉\vrule \@width \@wholewidth
407 〈latexrelease〉\@height \@dashdim\put(0,0){\copy\@dashbox}%
408 〈latexrelease〉\put(#2,0){\copy\@dashbox}%
409 〈latexrelease〉\put(0,#3){\lower\@dashdim\copy\@dashbox}%
410 〈latexrelease〉\put(#2,#3){\lower\@dashdim\copy\@dashbox}%
411 〈latexrelease〉\multiply\@dashdim \thr@@
412 〈latexrelease〉\fi
413 〈latexrelease〉\setbox\@dashbox\hbox{\vrule \@width \@wholewidth
414 〈latexrelease〉\@height #1\unitlength}\@tempcnta\z@
415 〈latexrelease〉\put(0,0){%
416 〈latexrelease〉 \hskip -\@halfwidth \vbox{\@whilenum \@tempcnta <\@dashcnt
417 〈latexrelease〉 \do{\vskip #1\unitlength\copy\@dashbox
418 〈latexrelease〉 \advance\@tempcnta\@ne }%
419 〈latexrelease〉 \vskip\@dashdim}\@tempcnta\z@
420 〈latexrelease〉\put(#2,0){%
421 〈latexrelease〉 \hskip -\@halfwidth \vbox{\@whilenum \@tempcnta<\@dashcnt
422 〈latexrelease〉 \do{\vskip #1\unitlength\copy\@dashbox
423 〈latexrelease〉 \advance\@tempcnta \@ne }%
424 〈latexrelease〉 \vskip\@dashdim}\}@\makepicbox(#2,#3)}
425 〈latexrelease〉\EndIncludeInRelease
426 〈*2ekernel〉

```

(End definition for \dashbox.)

*Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*  
CIRCLES AND OVALS

#### USER COMMANDS:

\circle{D} : Produces the circle with the diameter as close as possible to D \* \unitlength. \put(X,Y){\circle{D}} puts the circle with its center at (X,Y).

\oval(X,Y) : Makes an oval as round as possible that fits in the rectangle of width X \* \unitlength and height Y \* \unitlength. The reference point is the center.

\oval(X,Y)[POS] : Save as \oval(X,Y) except it draws only the half or quadrant of the oval indicated by POS. E.G., \oval(X,Y)[t] draws just the top half and \oval(X,Y)[br] draws just the bottom right quadrant. In all cases, the reference point is the same as the unqualified \oval(X,Y) command.

\@ovvert {DELTA1} {DELTA2} : Makes a vbox containing either the left side or the right side of the oval being constructed. The baseline will coincide with the outside bottom edge of the oval; the left side of the box will coincide with the left edge of the vertical

rule. The width of the box will be `\@tempdima`.  
 DELTA1 and DELTA2 are added to the character number in `\@tempcnta` to get the characters for the top and bottom quarter circle pieces.

`\@ovhorz` : Makes an hbox containing the straight rule for either the top or the bottom of the oval being constructed. The baseline will coincide with bottom edge of the rule; the left side of the box will coincide with the left side of the oval.  
 The width of the box will be `\@ovxx`.

`\@getcirc {DIAM}` : Sets `\@tempcnta` to the character number of the top-right quarter circle with the largest diameter less than or equal to DIAM.  
 Sets `\@tempboxa` to an hbox containing that character.  
 Sets `\@tempdima` to `\wd \@tempboxa`, which is the distance from the circle's left outside edge to its right inside edge.  
 (These characters are like those described in the TeXbook, pp. 389-90.)

```
\@getcirc {DIAM} ==
BEGIN
 \@tempcnta := integer coercion of (DIAM + 2pt)
 + 2pt added 1 Nov 88
 \@tempcnta := \@tempcnta / integer coercion of 4pt
 if \@tempcnta > 10
 then \@tempcnta := 10 fi
 if \@tempcnta > 0
 then \@tempcnta := \@tempcnta-1
 else LaTeX Warning: Oval too small.
 fi
 \@tempcnta := 4 * \@tempcnta
 \@tempboxa := \hbox{\@circlefnt \char \@tempcnta}
 \@tempdima := \wd \@tempboxa
END

\@put{X}{Y}{OBJ} ==
BEGIN
 \raise Y \hb@xt@ 0pt{\hskip X OBJ \hss}
END

\@oval(X,Y)[POS] ==
BEGIN
 \begingroup
 \boxmaxdepth := \maxdimen
 @ovt := @ovb := @ovl := @ovr := true
 for all E in POS
 do @ovE := false od
 \@ovxx := X * \unitlength
 \@ovyy := Y * \unitlength
```

```

\@tempdimb := min(\@ovxx,\@ovyy)
\@getcirc{\@tempdimb-2pt} %% "-2pt" added 7 Dec 89
\@ovro := \ht \@tempboxa
\@ovri := \dp \@tempboxa
\@ovdx := \ovxx - \@tempdima
\@ovdx := \@ovdx/2
\@ovdy := \ovyy - \@tempdima
\@ovdy := \ovyy/2
\@circlefnt
\@tempboxa :=
\hbox{
 if @ovr
 then \@ovvert{3}{2} \kern -\@tempdima
 fi
 if @ovl
 then \kern \ovxx \@ovvert{0}{1} \kern -\@tempdima
 \kern -\ovxx
 fi
 if @ovt
 then \ovhorz \kern -\ovxx
 fi
 if @ovb
 then \raise \ovyy \ovhorz
 fi
}
\@ovdx := \@ovdx + \@ovro
\@ovdy := \@ovdy + \@ovro
\ht\@tempboxa := \dp\@tempboxa := 0
\@put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}
\endgroup
END

\@ovvert {DELTA1} {DELTA2} ==
BEGIN
 \vbox to \ovyy {
 if @ovb
 then \tempcntb := \tempcnta + DELTA1
 \kern -\ovro
 \hbox { \char \tempcntb }
 \nointerlineskip
 else \kern \ovri \kern \ovdy
 fi
 \leaders \vrule width \wholewidth \vfil
 \nointerlineskip
 if @ovt
 then \tempcntb := \tempcnta + DELTA2
 \hbox { \char \tempcntb }
 else \kern \ovdy \kern \ovro
 fi
 }

```

```

END

\@ovhorz ==
BEGIN
\hb@xt@ \@ovxx{
 \kern \@ovro
 if @ovr
 then
 else \kern \@ovdx
 fi
 \leaders \hrule height \@wholewidth \hfil
 if @ovl
 then
 else \kern \@ovdx
 fi
 \kern \@ovri
}
END

\circle{DIAM} ==
BEGIN
\begingroup
\boxmaxdepth := maxdimen
\@tempdimb := DIAM *\unitlength
if \@tempdimb > 15.5pt
 then \@getcirc{\@tempdimb}
 \@ovro := \ht \tempboxa
 \tempboxa := \hbox{
 \circleft
 \tempcpta := \tempcpta + 2
 \char \tempcpta
 \tempcpta := \tempcpta - 1
 \char \tempcpta
 \kern -2\tempdima
 \tempcpta := \tempcpta + 2
 \raise \tempdima \hbox { \char \tempcpta }
 \raise \tempdima \box\tempboxa
 }
 \ht\tempboxa := \dp\tempboxa := 0
 \put{-\ovro}{-\ovro}{\tempboxa}
 else
 \circ{\@tempdimb}{96}
 fi
\endgroup
END

\circle*{DIAM} == \dot{DIAM} == \circ{DIAM*\unitlength}{112}

\circ{DIAM}{CHAR} ==
BEGIN

```

```

\@tempcnta := integer coercion of (DIAM + .5pt)/1pt.
if \@tempcnta > 15 then \@tempcnta := 15 fi
if \@tempcnta > 1 then \@tempcnta := \@tempcnta - 1 fi
\@tempcnta := \@tempcnta + CHAR
\@circlefnt
\char \@tempcnta
END

```

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

```

\if@ovt If producing the Top Bottom Left or Right of an oval.
\if@ovb 427 \newif\if@ovt
\if@owl 428 \newif\if@ovb
\if@ovr 429 \newif\if@owl
430 \newif\if@ovr

```

*(End definition for \if@ovt and others.)*

```

\@ovxx
\@ovyy 431 \newdimen\@ovxx
\@ovdx 432 \newdimen\@ovyy
\@ovdy 433 \newdimen\@ovdx
\@ovro 434 \newdimen\@ovdy
\@ovri 435 \newdimen\@ovro
436 \newdimen\@ovri

```

*(End definition for \@ovxx and others.)*

\advance\@tempdima 2pt\relax added 1 Nov 88 to fix bug in which size of drawn circle not monotonic function of argument of \circle, caused by different rounding for dimensions of large and small circles.

```

\@getcirc
437 \def\@getcirc#1{\@tempdima #1\relax \advance\@tempdima 2\p@
438 \@tempcnta\@tempdima
439 \@tempdima 4\p@\divide\@tempcnta\@tempdima
440 \ifnum \@tempcnta >10\relax
441 \@picture@warn
442 \@tempcnta 10\relax
443 \fi
444 \ifnum \@tempcnta >\z@ \advance\@tempcnta\m@ne
```

Warn if requirements for oval or circle can't be met.

```

445 \else \@picture@warn \fi
446 \multiply\@tempcnta 4\relax
447 \setbox\@tempboxa \hbox{\@circlefnt
448 \char\@tempcnta}\@tempdima \wd\@tempboxa}
```

*(End definition for \@getcirc.)*

\@picture@warn Generic warning for lines, vectors (used in \@sline) and oval or circle (used in \@getcirc) are not available at right size.

```

449 \def\@picture@warn{\@latex@warning{%
450 \string\oval, \string\circle, or \string\line\space
451 size unavailable}}
```

(End definition for \@picture@warn.)

```
\@put
452 \def\@put#1#2#3{\raise #2\hb@xt@z@{\hskip #1#3\hss}}
```

(End definition for \@put.)

```
\oval
453 \def\oval(#1,#2){\@ifnextchar[{\@oval(#1,#2)}{\@oval(#1,#2)[]}}

(End definition for \oval.)

454 </2ekernel>
455 <latexrelease>\IncludeInRelease{2016/03/31}%
456 <latexrelease> {\@ovhlinetrue} %
457 <latexrelease> {Avoid almost zero length leaders} %
458 <2ekernel | latexrelease>
```

\if@ovvline Tests whether horizontal or vertical lines are needed.

```
\if@ovhline
459 \newif\if@ovvline \@ovvlinetrue
460 \newif\if@ovhline \@ovhlinetrue
461 % \begin{macrocode}
462 </2ekernel | latexrelease>
463 <latexrelease>\EndIncludeInRelease
464 <latexrelease>\IncludeInRelease{0000/00/00}%
465 <latexrelease> {\@ovhlinetrue} %
466 <latexrelease> {Avoid almost zero length leaders} %
467 <latexrelease>\let\if@ovvline\@undefined
468 <latexrelease>\let\if@ovhline\@undefined
469 <latexrelease>\EndIncludeInRelease
470 <2ekernel>
```

(End definition for \if@ovvline and \if@ovhline.)

```
\oval
471 </2ekernel>
472 <2ekernel | latexrelease>
473 <latexrelease>\IncludeInRelease{2020/10/01}%
474 <latexrelease> {\@oval}{default units} %
475 \def\@oval(#1,#2)[#3]{\begingroup\boxmaxdepth \maxdimen
476 \@ovttrue \@ovbtrue \@ovltrue \@ovrtrue

477 \@ovvlinefalse \@ovhlinefalse
478 \atfor\reserved@a :=#3\do{ %
479 \csname @ov\reserved@a false\endcsname} %
480 \defaultunitsset\@ovxx{#1}\unitlength
481 \defaultunitsset\@ovyy{#2}\unitlength

482 \tempdima \ifdim \@ovyy >\@ovxx \@ovxx \@ovvlinetrue
483 \else \@ovyy \ifdim \@ovyy =\@ovxx \else \@ovhlinetrue \fi\fi
484 \advance \tempdima -2\p@
485 \getcirc \tempdima
486 \ovro \ht\tempboxa \ovri \dp\tempboxa
487 \ovdx\@ovxx \advance\ovdx -\tempdima \divide\ovdx \tw@
488 \ovdy\@ovyy \advance\ovdy -\tempdima \divide\ovdy \tw@
```

```

489 \ifdim \@ovdx >\z@ \@ovhlinetrue \fi
490 \ifdim \@ovdy >\z@ \@ovvlinetrue \fi
491 \circlefnt \setbox\@tempboxa
492 \hbox{\if@vr \@ovvert32\kern -\@tempdima \fi
493 \if@vl \kern \@ovxx \@ovvert01\kern -\@tempdima \kern -\@ovxx \fi
494 \if@vt \@ovhorz \kern -\@ovxx \fi
495 \if@vb \raise \@ovyy \@ovhorz \fi}\advance\@ovdx\@ovro
496 \advance\@ovdy\@ovro \ht\@tempboxa\z@ \dp\@tempboxa\z@
497 \put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}%
498 \endgroup
499 (//ekernel | latexrelease)

500 \EndIncludeInRelease
501 \IncludeInRelease{2016/03/31}%
502 \oval{[\@oval]{default units}}%
503 \def\oval(#1,#2)[#3]{\begin{group}\boxmaxdepth \maxdimen
504 \ovtrue \ovbtrue \ovltrue \ovrtrue
505 \ovlinefalse \ovhlinefalse
506 \tfor\reserved@a :=#3\do{%
507 \csname \ov\reserved@a false\endcsname}%
508 \ovxx #1\unitlength
509 \ovyy #2\unitlength
510 \tempdimb \ifdim \ovyy >\ovxx \ovxx \ovvlinetrue
511 \else \ovyy \ifdim \ovyy =\ovxx \else \ovhlinetrue
512 \fi\fi
513 \advance \tempdimb -2\p@
514 \getcirc \tempdimb
515 \ovro \ht\@tempboxa \ovri \dp\@tempboxa
516 \ovdx\ovxx \advance\ovdx -\tempdima \divide\ovdx \tw@
517 \ovdy\ovyy \advance\ovdy -\tempdima \divide\ovdy \tw@
518 \ifdim \ovdx >\z@ \ovhlinetrue \fi
519 \ifdim \ovdy >\z@ \ovvlinetrue \fi
520 \circlefnt \setbox\@tempboxa
521 \hbox{\if@vr \@ovvert32\kern -\@tempdima \fi
522 \if@vl
523 \kern \@ovxx \@ovvert01\kern -\@tempdima \kern -\@ovxx
524 \fi
525 \if@vt \@ovhorz \kern -\@ovxx \fi
526 \if@vb \raise \@ovyy \@ovhorz \fi}\advance\@ovdx\@ovro
527 \advance\@ovdy\@ovro \ht\@tempboxa\z@ \dp\@tempboxa\z@
528 \put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}%
529 \endgroup
530 \EndIncludeInRelease

531 \IncludeInRelease{0000/00/00}%
532 \oval{[\@oval]{default units}}%
533 \def\oval(#1,#2)[#3]{\begin{group}\boxmaxdepth \maxdimen
534 \ovtrue \ovbtrue \ovltrue \ovrtrue
535 \tfor\reserved@a :=#3\do
536 \csname \ov\reserved@a false\endcsname}%
537 \ovxx #1\unitlength
538 \ovyy #2\unitlength
539 \tempdimb \ifdim \ovyy >\ovxx \ovxx\else \ovyy \fi
540 \advance \tempdimb -2\p@
541 \getcirc \tempdimb

```

```

542 <|latexrelease> \@ovro \ht\@tempboxa \@ovri \dp\@tempboxa
543 <|latexrelease> \@ovdx\@ovxx \advance\@ovdx -\@tempdima \divide\@ovdx \tw@
544 <|latexrelease> \@ovdy\@ovyy \advance\@ovdy -\@tempdima \divide\@ovdy \tw@
545 <|latexrelease> \@circlefnt \setbox\@tempboxa
546 <|latexrelease> \hbox{\if@ovr \@ovvert32\kern -\@tempdima \fi
547 <|latexrelease> \if@ovl
548 <|latexrelease> \kern \@ovxx \@ovvert01\kern -\@tempdima \kern -\@ovxx
549 <|latexrelease> \fi
550 <|latexrelease> \if@ovt \@ovhorz \kern -\@ovxx \fi
551 <|latexrelease> \if@ovb \raise \@ovyy \@ovhorz \fi}\advance\@ovdx\@ovro
552 <|latexrelease> \advance\@ovdy\@ovro \ht\@tempboxa\z@\dp\@tempboxa\z@
553 <|latexrelease> \@put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}%
554 <|latexrelease> \endgroup
555 <|latexrelease>\EndIncludeInRelease
556 <|2ekernel>

```

(End definition for \@oval.)

### \@ovvert

```

557 <|2ekernel>
558 <|latexrelease>\IncludeInRelease{2016/03/31}%
559 <|latexrelease> {\@ovvert}{Avoid almost zero length leaders}%
560 <|2ekernel | latexrelease>
561 \def\@ovvert#1#2{\vbox to\@ovyy{%
562 \if@ovb \@tempcntb \@tempcnta \advance \@tempcntb #1\relax
563 \kern -\@ovro \hbox{\char \@tempcntb}\nointerlineskip
564 \else \kern \@ovri \kern \@ovdy \fi
565 \if@ovvline \leaders\vrule \@width \@wholewidth \fi
566 \vfil \nointerlineskip
567 \if@ovt \@tempcntb \@tempcnta \advance \@tempcntb #2\relax
568 \hbox{\char \@tempcntb}%
569 \else \kern \@ovdy \kern \@ovro \fi}
570 <|2ekernel | latexrelease>
571 <|latexrelease>\EndIncludeInRelease
572 <|latexrelease>\IncludeInRelease{0000/00/00}%
573 <|latexrelease> {\@ovvert}{Avoid almost zero length leaders}%
574 <|latexrelease>\def\@ovvert#1#2{\vbox to\@ovyy{%
575 \if@ovb \@tempcntb \@tempcnta \advance \@tempcntb #1\relax
576 \kern -\@ovro \hbox{\char \@tempcntb}\nointerlineskip
577 \else \kern \@ovri \kern \@ovdy \fi
578 \leaders\vrule \@width \@wholewidth\vfil \nointerlineskip
579 \if@ovt \@tempcntb \@tempcnta \advance \@tempcntb #2\relax
580 \hbox{\char \@tempcntb}%
581 \else \kern \@ovdy \kern \@ovro \fi}
582 <|latexrelease>\EndIncludeInRelease
583 <|2ekernel>

```

(End definition for \@ovvert.)

### \@ovhorz

```

584 <|2ekernel>
585 <|latexrelease>\IncludeInRelease{2016/03/31}%
586 <|latexrelease> {\@ovhorz}{Avoid almost zero length leaders}%

```

```

587 {*2ekernel | latexrelease}
588 \def\@ovhorz{\hb@xt@0@ovxx{\kern \c@ovro
589 \if@ovr \else \kern \c@ovdx \fi
590 \if@ovhline \leaders \hrule \c@height \c@wholewidth \fi
591 \hfil
592 \if@ovl \else \kern \c@ovdx \fi
593 \kern \c@ovri}}
594 {/2ekernel | latexrelease}
595 \end{IncludeInRelease}
596 \IncludeInRelease{0000/00/00}%
597 \end{latexrelease} {\c@ovhorz}{Avoid almost zero length leaders}%
598 \def\@ovhorz{\hb@xt@0@ovxx{\kern \c@ovro
599 \if@ovr \else \kern \c@ovdx \fi
600 \leaders \hrule \c@height \c@wholewidth \hfil
601 \if@ovl \else \kern \c@ovdx \fi
602 \kern \c@ovri}}
603 \end{IncludeInRelease}
604 {*2ekernel}

(End definition for \c@ovhorz.)
```

\circle

```

605 \def\circle{\c@inmatherr\circle\c@ifstar\c@dot\c@circle}
```

(End definition for \circle.)

\@circle

```

606 {/2ekernel}
607 {*2ekernel | latexrelease}
608 \end{latexrelease} \IncludeInRelease{2020/10/01}%
609 \end{latexrelease} {\c@circle}{default units}%
610 \def\@circle#1{%
611 \begingroup \boxmaxdepth \maxdimen
612 \c@defaultunitsset\c@tempdimb{#1}\unitlength
613 \ifdim \c@tempdimb >15.5\p@ \getcirc\c@tempdimb
614 \c@ovro\ht\c@tempboxa
615 \setbox\c@tempboxa\hbox{\c@circlefont
616 \advance\c@tempcnta\tw@ \char \c@tempcnta
617 \advance\c@tempcnta\m@ne \char \c@tempcnta \kern -2\c@tempdima
618 \advance\c@tempcnta\tw@
619 \raise \c@tempdima \hbox{\char\c@tempcnta}\raise \c@tempdima
620 \box\c@tempboxa\ht\c@tempboxa\z@ \dp\c@tempboxa\z@
621 \c@put{-\c@ovro}{-\c@ovro}{\box\c@tempboxa}%
622 \else \c@circ\c@tempdimb{96}\fi\endgroup
623 {/2ekernel | latexrelease}
624 \end{IncludeInRelease}
625 \IncludeInRelease{0000/00/00}%
626 \end{latexrelease} {\c@circle}{default units}%
627 \def\@circle#1{%
628 \begingroup \boxmaxdepth \maxdimen \c@tempdimb #1\unitlength
629 \ifdim \c@tempdimb >15.5\p@ \getcirc\c@tempdimb
630 \c@ovro\ht\c@tempboxa
```

```

631 <|latexrelease> \setbox\@tempboxa\hbox{\@circlefnt
632 <|latexrelease> \advance\@tempcnta\tw@ \char \@tempcnta
633 <|latexrelease> \advance\@tempcnta\m@ne \char \@tempcnta
634 <|latexrelease> \kern -2\@tempdima
635 <|latexrelease> \advance\@tempcnta\tw@
636 <|latexrelease> \raise \@tempdima \hbox{\char\@tempcnta}%
637 <|latexrelease> \raise \@tempdima
638 <|latexrelease> \box\@tempboxa\ht\@tempboxa\z@ \dp\@tempboxa\z@
639 <|latexrelease> \put{-\@ovro}{-\@ovro}{\box\@tempboxa}%
640 <|latexrelease> \else \circ\@tempdimb{96}\fi\endgroup}
641 <|latexrelease>\EndIncludeInRelease
642 <|2ekernel>

```

(End definition for `\@circle`.)

`\@dot` Internal form of `\circle*`.

```

643 </2ekernel>
644 <|2ekernel | latexrelease>
645 <|latexrelease>\IncludeInRelease{2020/10/01}%
646 <|latexrelease> {\@dot}{default units}%
647 \def\@dot#1{%
648 \@defaultunitsset\@tempdimb{#1}\unitlength
649 \circ\@tempdimb{112}%
650 </2ekernel | latexrelease>
651 <|latexrelease>\EndIncludeInRelease
652 <|latexrelease>\IncludeInRelease{0000/00/00}%
653 <|latexrelease> {\@dot}{default units}%
654 <|latexrelease>\def\@dot#1{\@tempdimb #1\unitlength \circ\@tempdimb{112}%
655 <|latexrelease>\EndIncludeInRelease
656 <|2ekernel>

```

(End definition for `\@dot`.)

`\@circ`

```

657 \def\@circ#1#2{\@tempdima #1\relax \advance\@tempdima .5\p@
658 \atempcnta\@tempdima \atempdima \p@
659 \divide\atempcnta\@tempdima
660 \ifnum\atempcnta >15\relax \atempcnta 15\relax \fi
661 \ifnum \atempcnta >\z@ \advance\atempcnta\m@ne\fi
662 \advance\atempcnta #2\relax
663 \circlefnt \char\atempcnta}

```

(End definition for `\@circ`.)

`\@xarg` Counters used for manipulating the ‘slope’ arguments.

```

664 \newcount\@xarg
665 \newcount\@yarg
666 \newcount\@yyarg

```

(End definition for `\@xarg`, `\@yarg`, and `\@yyarg`.)

`\@multicnt` Counter used in `\multiput`, and also `\multicolumn`.

```

667 \newcount\@multicnt

```

(End definition for `\@multicnt`.)

```

\@xdim Length registers.
\@ydim 668 \newdimen\@xdim
669 \newdimen\@ydim

(End definition for \@xdim and \@ydim.)

\@linechar Box for holding a line segment character, for sloping lines.
670 \newbox\@linechar

(End definition for \@linechar.)

\@linelen Length of the line currently being built.
671 \newdimen\@linelen

(End definition for \@linelen.)

\@clnwd Height and width of current line segment.
\@clnht 672 \newdimen\@clnwd
673 \newdimen\@clnht

(End definition for \@clnwd and \@clnht.)

\@dashdim \dashbox internal registers.
\@dashbox 674 \newdimen\@dashdim
\@dashcnt 675 \newbox\@dashbox
676 \newcount\@dashcnt

(End definition for \@dashdim, \@dashbox, and \@dashcnt.)
Initialization: “\thinlines”
677 \let\@linefnt\tenln
678 \let\@circlefnt\tencirc
679 \wholewidth\fontdimen8\tenln
680 \halfwidth .5\wholewidth

```

## 1.1 Curves

The new `\qbezier` command, based on the old `\bezier` defined in `bezier.sty`.  
*Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*

```

\qbezier[N] == \bezier{N}

\bezier{N}(AX,AY)(BX,BY)(CX,CY) ==
BEGIN
 IF N = 0
 THEN \@xima := |BX - AX|
 \@xb := |CX - BX|
 \@xa := Max(\@xa, \@xb)
 \@ya := |BY - AY|
 \@yb := |CY - BY|
 \@ya := Max(\@ya, \@yb)
 @sc := Max(\@xa, \@ya)
 %% The coefficient .5 below is the degree of overlap of
 %% successive points, where 1 is no overlap and 0 is

```

```

%% complete overlap. A coefficient of C multiplies
%% the number of points plotted by 1/C.
%%
\@xa := .5 * \halfwidth
@sc := @sc / \halfwidth
@sc := Max(@sc, qbeziermax)
ELSE @sc := N
@scp := @sc+1
\@xb := 2 * (BX - AX) * \unitlength
\@xa := ((CX-AX)*\unitlength - \@xb)/@sc
\@yb := 2 * (BY - AY) * \unitlength
\@ya := ((CY-AY)*\unitlength - \@yb)/@sc
\@pictdot := square rule of width \wholewidth
\count@ := 0
WHILE \count@ < @scp
 DO \@xdim := ((\count@*\@xa + @xb) / @sc) * \count@
 \@ydim := ((\count@*\@ya + @yb) / @sc) * \count@
 plot pt with relative coords (\@xdim,\@ydim)
 \count@ := \count@+1
OD

```

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

\qbeziermax The maximum number of points to plot.

681 \def\qbeziermax{500}

(*End definition for \qbeziermax.*)

In the code below, to save registers \@a ... are not used. Instead other registers are reused.

```

\newcounter{@sc} -> \c@multicnt
\newcounter{@scp} -> \@tempcnta
\newdimen\@xa -> \@ovxx
\newdimen\@xb -> \@ovdx
\newdimen\@ya -> \@ovyy
\newdimen\@yb -> \@ovdy
\newsavebox{\@pictdot} -> \@tempboxa

```

\qbezier Main user-level command to plot quadratic bezier curves. #2 should be (.

682 \newcommand\qbezier[2][0]{\bezier{#1}{#2}}

(*End definition for \qbezier.*)

\bezier Form of \bezier compatible with 2.09 *bezier.sty*, but modified to ignore spaces between its arguments. #2 should be white space, and #4 should be (.

683 \def\bezier#1#2(#3)#4({\@bezier#1)(#3)()}

```

\@bezier 684 </2ekernel>
685 <*2ekernel | latexrelease>
686 <latexrelease>\IncludeInRelease{2020/10/01}%
687 <latexrelease> {\@bezier}{default units}%
688 \def\@bezier#1(#2,#3)(#4,#5)(#6,#7){%
689 \ifnum #1=\z@
690 \@defaultunitsset{@ovxx{#4}\unitlength
691 \@defaultunitsset{\advance{@ovxx}{-#2}\unitlength
692 \ifdim \@ovxx<\z@ \@ovxx -\@ovxx \fi
693 \@defaultunitsset{@ovdx{#6}\unitlength
694 \@defaultunitsset{\advance{@ovdx}{-#4}\unitlength
695 \ifdim \@ovdx<\z@ \@ovdx -\@ovdx \fi
696 \ifdim \@ovxx<\@ovdx \@ovxx \@ovdx \fi
697 \@defaultunitsset{@ovyy{#5}\unitlength
698 \@defaultunitsset{\advance{@ovyy}{-#3}\unitlength
699 \ifdim \@ovyy<\z@ \@ovyy -\@ovyy \fi
700 \@defaultunitsset{@ovdy{#7}\unitlength
701 \@defaultunitsset{\advance{@ovdy}{-#5}\unitlength
702 \ifdim \@ovdy<\z@ \@ovdy -\@ovdy \fi
703 \ifdim \@ovyy<\@ovdy \@ovyy \@ovdy \fi
704 \@multicnt
705 \ifdim \@ovxx>\@ovyy \@ovxx \else \@ovyy \fi
706 \@ovxx .5\@halfwidth \divide{@multicnt}{@ovxx}
707 \ifnum \qbeziermax<\@multicnt
708 \@multicnt\qbeziermax\relax
709 \fi
710 \else \@multicnt#1\relax \fi
711 \@tempcpta\@multicnt \advance{@tempcpta}{one}
712 \@defaultunitsset{@ovdx{#4}\unitlength
713 \@defaultunitsset{\advance{@ovdx}{-#2}\unitlength
714 \multiply{@ovdx}{tw@}
715 \@defaultunitsset{@ovxx{#6}\unitlength
716 \@defaultunitsset{\advance{@ovxx}{-#2}\unitlength
717 \advance{@ovxx}{-}\@ovdx \divide{@ovxx}{@multicnt}
718 \@defaultunitsset{@ovdy{#5}\unitlength
719 \@defaultunitsset{\advance{@ovdy}{-#3}\unitlength
720 \multiply{@ovdy}{tw@}
721 \@defaultunitsset{@ovyy{#7}\unitlength
722 \@defaultunitsset{\advance{@ovyy}{-#3}\unitlength
723 \advance{@ovyy}{-}\@ovdy \divide{@ovyy}{@multicnt

724 \setbox{@tempboxa}\hbox{%
725 \hspace{-}\@halfwidth
726 \vrule \height\@halfwidth
727 \depth \@halfwidth
728 \width \wholewidth}%
729 \put(#2,#3){%
730 \count@\z@
731 \whilenum{\count@<\@tempcpta}\do
732 {\@xdim\count@\@ovxx
733 \advance{@xdim}{\@ovdx}
734 \divide{@xdim}{@multicnt}
735 \multiply{@xdim}{\count@}

```

```

736 \@ydim\count@\@ovyy
737 \advance\@ydim\@ovdy
738 \divide\@ydim\@multicnt
739 \multiply\@ydim\count@
740 \raise \@ydim
741 \hb@xt@\z@{\kern\@xdim
742 \unhcopy\@tempboxa\hss}%
743 \advance\count@\@ne}}}
744 /{2ekernel | latexrelease}

745 \end{IncludeInRelease}
746 \IncludeInRelease{0000/00/00} %
747 \bezier {default units}%
748 \def\bezier#1(#2,#3)(#4,#5)(#6,#7){%
749 \ifnum #1=\z@
750 \@ovxx #4\unitlength
751 \advance\@ovxx -#2\unitlength
752 \ifdim \@ovxx<\z@ \@ovxx -\@ovxx \fi
753 \@ovdx #6\unitlength
754 \advance\@ovdx -#4\unitlength
755 \ifdim \@ovdx<\z@ \@ovdx -\@ovdx \fi
756 \ifdim \@ovxx<\@ovdx \@ovxx \@ovdx \fi
757 \@ovyy #5\unitlength
758 \advance\@ovyy -#3\unitlength
759 \ifdim \@ovyy<\z@ \@ovyy -\@ovyy \fi
760 \@ovdy #7\unitlength
761 \advance\@ovdy -#5\unitlength
762 \ifdim \@ovdy<\z@ \@ovdy -\@ovdy \fi
763 \ifdim \@ovyy<\@ovdy \@ovyy \@ovdy \fi
764 \@multicnt
765 \ifdim \@ovxx>\@ovyy \@ovxx \else \@ovyy \fi
766 \@ovxx .5\@halfwidth \divide\@multicnt\@ovxx
767 \ifnum
768 \qbezier{max}{\@multicnt \@multicnt\qbezier{max}\relax}
769 \fi
770 \else \@multicnt#1\relax \fi
771 \@tempcnta\@multicnt \advance\@tempcnta\@ne
772 \@ovdx #4\unitlength \advance\@ovdx -#2\unitlength
773 \multiply\@ovdx \tw@
774 \@ovxx #6\unitlength \advance\@ovxx -#2\unitlength
775 \advance\@ovxx -\@ovdx \divide\@ovxx\@multicnt
776 \@ovdy #5\unitlength \advance\@ovdy -#3\unitlength
777 \multiply\@ovdy \tw@
778 \@ovyy #7\unitlength \advance\@ovyy -#3\unitlength
779 \advance\@ovyy -\@ovdy \divide\@ovyy\@multicnt
780 \setbox\@tempboxa\hbox{%
781 \hskip -\@halfwidth
782 \vrule \height\@halfwidth
783 \depth \@halfwidth
784 \width \@wholewidth} %
785 \put{#2,#3}{%
786 \count@\z@
787 \whilenum{\count@\l\@tempcnta}\do
788 {\@xdim\count@\@ovxx
789 \advance\@xdim\@ovdx

```

```

790 \divide\@xdim\@multicnt
791 \multiply\@xdim\count@
792 \ydim\count@\@ovyy
793 \advance\ydim\@ovdy
794 \divide\ydim\@multicnt
795 \multiply\ydim\count@
796 \raise\@ydim
797 \hb@xt@z@{\kern\@xdim
798 \unhcopy\tempboxa\hss}%
799 \advance\count@\@ne}}}
800 \EndIncludeInRelease
801 {*2ekernel}

```

(End definition for `\bezier` and `\qbezier`.)

As the commands above all use “picture” interface we couldn’t define them with `\DeclareRobustCommand` so we do that now.

```

802
```

- 803 {\*2ekernel | latexrelease}
- 804 \IncludeInRelease{2019/10/01}%
- 805 \bezier{}{Make commands robust}%
- 806 \MakeRobust\bezier
- 807 \MakeRobust\circle
- 808 \MakeRobust\dashbox
- 809 \MakeRobust\line
- 810 \MakeRobust\linethickness
- 811 \MakeRobust\multiput
- 812 \MakeRobust\oval
- 813 \MakeRobust\put
- 814 \MakeRobust\qbezier
- 815 \MakeRobust\shortstack
- 816 \MakeRobust\thinlines
- 817 \MakeRobust\vector
- 818
- 819 \EndIncludeInRelease
- 820 \IncludeInRelease{0000/00/00}%
- 821 \bezier{}{Make commands robust}%
- 822
- 823 \kernel@make@fragile\bezier
- 824 \kernel@make@fragile\circle
- 825 \kernel@make@fragile\dashbox
- 826 \kernel@make@fragile\line
- 827 \kernel@make@fragile\linethickness
- 828 \kernel@make@fragile\multiput
- 829 \kernel@make@fragile\oval
- 830 \kernel@make@fragile\put
- 831 \kernel@make@fragile\qbezier
- 832 \kernel@make@fragile\shortstack
- 833 \kernel@make@fragile\thinlines
- 834 \kernel@make@fragile\vector
- 835
- 836 \EndIncludeInRelease
- 837
- 838

# File N

## ltthm.dtx

### 1 Theorem Environments

The user creates his own theorem-like environments with the command

`\newtheorem{<name>}{<text>}[<counter>]` or  
`\newtheorem{<name>}[<oldname>]{<text>}`

This defines the environment `<name>` to be just as one would expect a theorem environment to be, except that it prints `<text>` instead of “Theorem”.

If `<oldname>` is given, then environments `<name>` and `<oldname>` use the same counter, so using a `<name>` environment advances the number of the next `<name>` environment, and vice-versa.

If `<counter>` is given, then environment `<name>` is numbered within `<counter>`.

E.g., if `<counter>` = `subsection`, then the first `<name>` in subsection 7.2 is numbered `<text> 7.2.1`.

The way `<name>` environments are numbered can be changed by redefining `\the<name>`. *Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*

#### DOCUMENT STYLE PARAMETERS

`\@thmcOUNTER` : A command such that

`\edef\theCOUNTER{\@thmcOUNTER}`

defines `\theCOUNTER` to produce a number for a theorem environment.

The default is:

`BEGIN \noexpand\arabic{COUNTER} END`

`\@thmcOUNTERsep` : A separator placed between a theorem number and the number of the counter within which it is numbered.

E.g., to make the third theorem of section 7.2 be numbered 7.2-3, `\@thmcOUNTERsep` should be `\def`'`. Its default is `"`.

`\@begintheorem{NAME}{NUMBER}` : A command that begins a theorem environment for a ‘theorem’ named ‘NAME NUMBER’ – e.g., `\@begintheorem{Lemma}{3.7}` starts Lemma 3.7.

`\@opargbegintheorem{NAME}{NUMBER}{OPARG}` :

A command that begins a theorem environment for a ‘theorem’ named ‘NAME NUMBER’ with optional argument OPARG – e.g., `\@begintheorem{Lemma}{3.7}{Jones}` starts ‘Lemma 3.7 (Jones):’.

`\@endtheorem` : A command that ends a theorem environment.

`\newtheorem{NAME}{TEXT}[COUNTER] ==`

`BEGIN`  
if `\NAME` is definable

```

then \@definecounter{NAME}
 if COUNTER present
 then \@newctr{NAME}[COUNTER] fi
 \theNAME == BEGIN \theCOUNTER \@thmcOUNTERsep
 eval\@thmcOUNTER{NAME} END
 else \theNAME == BEGIN eval\@thmcOUNTER{NAME} END
 \NAME == \@thm{NAME}{TEXT}
 \endNAME == \@endtheorem
 else error
 fi
 END

\newtheorem{NAME}[OLDNAME]{TEXT} ==
BEGIN
 if counter OLDNAME nonexistent
 then ERROR
 else
 if \NAME is definable
 then BEGIN
 \theNAME == \theOLDNAME
 \NAME == \@thm{OLDNAME}{TEXT}
 \endNAME == \@endtheorem
 END
 else error
 fi
END

\@thm{NAME}{TEXT} ==
BEGIN
 \refstepcounter{NAME}
 if next char =
 then \@ythm{NAME}{TEXT}
 else \@xthm{NAME}{TEXT}
 fi
END

\@xthm{NAME}{TEXT} ==
BEGIN
 \@begintheorem{TEXT}{\theNAME}
 \ignorespaces
END

\@ythm{NAME}{TEXT}[OPARG] ==
BEGIN
 \@opargbegintheorem{TEXT}{\theNAME}{OPARG}
 \ignorespaces
END

```

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

\newtheorem \newtheorem ought really be allowed only in the preamble. Which would be good document style, and allow some main memory to be saved by declaring these commands to be @onlypreamble. Unfortunately the L<sup>A</sup>T<sub>E</sub>X book indicates that \newtheorem may be used anywhere in the document...

```

1 {*2ekernel}
2 \def\newtheorem#1{%
3 \@ifnextchar[\{@othm{#1}\}{\@nthm{#1}}}

```

(End definition for \newtheorem.)

\@nthm

```

4 \def\@nthm#1#2{%
5 \@ifnextchar[\{@xnthm{#1}{#2}\}{\@ynthm{#1}{#2}}}

```

(End definition for \@nthm.)

\@xnthm 92/09/18 RmS: Changed \@addtoreset to \@newctr to produce error message if counter #3 does not exist (to be consistent with behaviour of \newcounter)

```

6 \def\@xnthm#1#2[#3]{%
7 \expandafter\@ifdefinable\csname #1\endcsname
8 {\@definecounter{#1}\@newctr{#1}[#3]\%
9 \expandafter\xdef\csname the#1\endcsname{\%
10 \expandafter\noexpand\csname the#3\endcsname \@thmcOUNTERsep
11 \@thmcOUNTER{#1}}\%
12 \global\@namedef{#1}{\@thm{#1}{#2}}\%
13 \global\@namedef{end#1}{\@endtheorem}}}

```

(End definition for \@xnthm.)

\@ynthm

```

14 \def\@ynthm#1#2{%
15 \expandafter\@ifdefinable\csname #1\endcsname
16 {\@definecounter{#1}\%
17 \expandafter\xdef\csname the#1\endcsname{\@thmcOUNTER{#1}}\%
18 \global\@namedef{#1}{\@thm{#1}{#2}}\%
19 \global\@namedef{end#1}{\@endtheorem}}}

```

(End definition for \@ynthm.)

\@othm

```

20 \def\@othm#1[#2]{#3}{%
21 \@ifundefined{c@#2}{\@nocounterr{#2}}\%
22 {\expandafter\@ifdefinable\csname #1\endcsname
23 {\global\@namedef{the#1}{\@nameuse{the#2}}\%
24 \global\@namedef{#1}{\@thm{#2}{#3}}\%
25 \global\@namedef{end#1}{\@endtheorem}}}

```

(End definition for \@othm.)

\@thm

```

26 \def\@thm#1#2{%
27 \refstepcounter{#1}\%
28 \@ifnextchar[\{@ythm{#1}{#2}\}{\@xthm{#1}{#2}}}

```

(End definition for \@thm.)

```

\@xthm
\@ythm 29 \def\@xthm#1#2{%
30 \begin{theorem}{\csname the#1\endcsname}\ignorespaces}
31 \def\@ythm#1#2[#3]{%
32 \opargbegintheorem{#2}{\csname the#1\endcsname}{#3}\ignorespaces}

(End definition for \@xthm and \@ythm.)

Default values

\@thmcnter
\@thmcntersep 33 \def\@thmcnter#1{\noexpand\arabic{#1}}
34 \def\@thmcntersep{.}

(End definition for \@thmcnter and \@thmcntersep.)

\begin{theorem} Providing theorem defaults.
\opargbegintheorem
\endtheorem 35 \def\begintheorem#1#2{\trivlist
36 \item[\hspace*{1em}\bfseries #1\, #2]\itshape}
37 \def\opargbegintheorem#1#2#3{\trivlist
38 \item[\hspace*{1em}\bfseries #1\, #2\, (#3)]\itshape}
39 \def\endtheorem{\endtrivlist}
40 \end{ekernel}

(End definition for \begintheorem, \opargbegintheorem, and \endtheorem.)

```

# File O

## ltsect.dtx

### 1 Sectioning Commands

This file defines the declarations such as `\author` which are used by `\maketitle`. `\maketitle` itself is defined by each class, not in the L<sup>A</sup>T<sub>E</sub>X kernel.

The second part of the file defines the generic commands used for defining sectioning commands such as `\chapter`. Again the actual document level commands are defined in the class files, in terms of these commands.

```
1 {*2ekernel}
2 \message{title,}
```

#### 1.1 The Title

`\title` The user defines the title and author by the declarations `\title{<name>}`, `\author{<name>}`.  
`\author` Similarly the date is declared with `\date{<date>}`.  
`\date` Inside these, the `\thanks{<footnote text>}` command may be used to make acknowledgements, notice of address, etc. in a footnote. If there are multiple authors, they have `\and` to be separated with the `\and` command.  
`\maketitle` And finally, the `\maketitle` command produces the actual title, using the information previously saved with the other commands.

```
3 /2ekernel
4 {*2ekernel | latexrelease}
5 {latexrelease}\IncludeInRelease{2019/10/01}%
6 {latexrelease} {\title}{Make commands robust}%
```

`\title` `\title` for use in `\maketitle`. If not given `\maketitle` will produce an error message.  
7 `\DeclareRobustCommand\title[1]{\gdef\@title{\#1}}`

(End definition for `\title`.)

`\author` `\author` for use in `\maketitle`. If not given `\maketitle` will produce a warning message.  
8 `\DeclareRobustCommand*\author[1]{\gdef\@author{\#1}}`

(End definition for `\author`.)

`\date` `\date` for use in `\maketitle`. If not given `\maketitle` will produce `\today` as the default.

```
9 \DeclareRobustCommand*\date[1]{\gdef\@date{\#1}}
```

(End definition for `\date`.)

`\thanks`  
10 `\DeclareRobustCommand\thanks[1]{\footnotemark}`  
11     `\protected\@xdef\@thanks{\@thanks`  
12         `\protect\footnotetext[\the\c@footnote]{\#1}}%`  
13 }

(End definition for `\thanks`.)

```

\and
14 \DeclareRobustCommand{\and}{%
15 \end{tabular}%
16 \hskip 1em \oplus .17fil%
17 \begin{tabular}[t]{c}}% \end{tabular}

(End definition for \and.)

18 </2ekernel | latexrelease>
19 <latexrelease>\EndIncludeInRelease
20 <latexrelease>\IncludeInRelease{0000/00/00}%
21 <latexrelease> {\title}{Make commands robust}%
22 <latexrelease>
23 <latexrelease>\kernel@make@fragile\title
24 <latexrelease>\kernel@make@fragile\author
25 <latexrelease>\kernel@make@fragile\date
26 <latexrelease>\kernel@make@fragile\thanks
27 <latexrelease>\kernel@make@fragile\and
28 <latexrelease>
29 <latexrelease>\EndIncludeInRelease
30 <*2ekernel>

\@title
31 \def\@title{\@latex@error{No \noexpand\title given}\@ehc}

(End definition for \@title.)

\@author
32 \def\@author{\@latex@warning@no@line{No \noexpand\author given}{}}

(End definition for \@author.)

\@date
33 \gdef\@date{\today}

(End definition for \@date.)

\@thanks
34 \let\@thanks\empty

(End definition for \@thanks.)

35 \message{sectioning,}


```

## 1.2 Sectioning

```

\@secpenalty
36 \newcount\@secpenalty
37 \@secpenalty = -300

(End definition for \@secpenalty.)

\if@noskipsec Way back in 1991 (08/26) FMi & RmS set the \@noskipsec switch to true for the
\@noskipsectrue preamble and to false in \document. This was done to trap lists and related text in the
preamble but it does not catch everything.
38 \newif\if@noskipsec \@noskipsectrue

```

(End definition for `\if@noskipsec` and `\@noskipsectrue`.)

`\@startsection` The `\@startsection{\name}{\level}{\indent}{\beforeskip}{\afterskip}{\style}` command is the mother of all the user level sectioning commands. The part after the `*`, including the `*` is optional.

**name:** e.g., 'subsection'

**level:** a number, denoting depth of section – e.g., chapter = 0, section = 1, etc.

**indent:** Indentation of heading from left margin

**beforekip:** Absolute value = skip to leave above the heading. If negative, then paragraph indent of text following heading is suppressed.

**afterskip:** if positive, then skip to leave below heading, else negative of skip to leave to right of run-in heading.

**style:** Commands to set style. Since June 1996 release the `last` command in this argument may be a command such as `\MakeUppercase` or `\fbox` that takes an argument. The section heading will be supplied as the argument to this command. So setting #6 to, say, `\bfseries\MakeUppercase` would produce bold, uppercase headings.

If '`*`' is missing, then increment the counter. If it is present, then there should be no `[altheading]` argument. The command uses the counter 'secnumdepth'. It contains a pointer to the highest section level that is to be numbered.

**Warning:** The `\@startsection` command should be at the same or higher grouping level as the text that follows it. For example, you should *not* do something like

```
\def\foo{ \begingroup ...
 \paragraph{...}
 \endgroup}
```

Pseudocode for the `\@startsection` command *Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments*  
(not necessarily accurate any more):

```
\@startsection
{NAME}{LEVEL}{INDENT}{BEFORESKIP}{AFTERSKIP}{STYLE} ==
BEGIN
 IF @noskipsec = T THEN \leavevmode FI
 % true if previous section had no body.

 \par
 @tempskipa := BEFORESKIP
 @afterindent := T
 IF @tempskipa < 0 THEN @tempskipa := -@tempskipa
 @afterindent := F
 FI
 IF @nobreak = true
 THEN \everypar == null
 ELSE \addpenalty{@secpenalty}
 \addvspace{@tempskipa}
 FI
 IF * next
```

```

 THEN \@ssect{INDENT}{BEForeskip}{AFTerskip}{Style}
 ELSE \@dblarg{\@sect
 {NAME}{LEVEL}{INDENT}
 {BEForeskip}{AFTerskip}{Style}}
 FI
END
End of historical LATEX 2.09 comments.

39 \def\@startsection#1#2#3#4#5#6{%
40 \if@noskipsec \leavevmode \fi
41 \par
42 \tempskipa #4\relax
43 \if@afterindenttrue
44 \ifdim \tempskipa <\z@
45 \tempskipa -\tempskipa \if@afterindentfalse
46 \fi
47 \if@nobreak
48 \everypar{}%
49 \else
50 \addpenalty\secpenalty\addvspace\tempskipa
51 \fi
52 \ifstar
53 {\@ssect{#3}{#4}{#5}{#6}}%
54 {\@dblarg{\@sect{#1}{#2}{#3}{#4}{#5}{#6}}}}

```

(End definition for \@startsection.)

\@sect Pseudocode for the \@sect command *Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more)*:

```

\@sect{NAME}{LEVEL}
 {INDENT}{BEForeskip}{AFTerskip}
 {Style}[ARG1][ARG2]
 ==
BEGIN
 IF LEVEL > \c@secnumdepth
 THEN \svsec :=L null
 ELSE \refstepcounter{NAME}
 \svsec :=L BEGIN \secntformat{#1}\relax END
 FI
 IF AFTERSKIP > 0
 THEN \begingroup
 Style
 \hangfrom{\hskip INDENT\svsec}
 {\interlinepenalty 10000 ARG2\par}
 \endgroup
 \NAMEmark{ARG1}
 \addcontentsline{toc}{NAME}
 { IF LEVEL > \c@secnumdepth
 ELSE \protect\numberline{\theNAME} FI
 ARG1 }
 ELSE \svsechd == BEGIN Style
 \hskip INDENT\svsec

```

```

ARG2
\NAMEmark{ARG1}
\addcontentsline{toc}{NAME}
{ IF LEVEL > \c@secnumdepth
ELSE
 \protect\numberline{\theNAME}
FI
ARG1 }

END
FI
\@xsect{AFTERSKIP}
END
End of historical LATEX 2.09 comments.
55 \def\@sect#1#2#3#4#5#6[#7]#8{%
56 \ifnum #2>\c@secnumdepth
57 \let\@svsec\@empty
58 \else
59 \refstepcounter{#1}%

```

Since \secformat might end with an improper \hskip which is scanning forward for plus or minus we end the definition of \@svsec with \relax as a precaution.

```

60 \protected@edef\@svsec{\secformat{#1}\relax}%
61 \fi
62 \tempskipa #5\relax
63 \ifdim \tempskipa>\z@
64 \begingroup

```

This { used to be after the argument to \changefrom but was moved here to allow commands such as \MakeUppercase to be used at the end of #6.

```

65 #6{%
66 \changefrom{\hskip #3\relax\@svsec}%
67 \interlinepenalty \OM #8\@@par}%
68 \endgroup
69 \csname #1mark\endcsname{#7}%
70 \addcontentsline{toc}{#1}{%
71 \ifnum #2>\c@secnumdepth \else
72 \protect\numberline{\csname the#1\endcsname}%
73 \fi
74 #7}%
75 \else
\relax added 2 May 90
76 \def\@svsechd{%
77 #6{\hskip #3\relax
78 \@svsec #8}%
79 \csname #1mark\endcsname{#7}%
80 \addcontentsline{toc}{#1}{%
81 \ifnum #2>\c@secnumdepth \else
82 \protect\numberline{\csname the#1\endcsname}%
83 \fi
84 #7}%
85 \fi
86 \@xsect{#5}%

```

(End definition for \@sect.)

\@xsect Pseudocode for the \@xsect command *Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*

```
\@xsect{AFTERSKIP} ==
BEGIN
 IF AFTERSKIP > 0
 THEN \par \nobreak
 \vskip AFTERSKIP
 \afterheading
 ELSE @nobreak :=G F
 @noskipsec :=G T
 \everypar{ IF @noskipsec = T
 THEN @noskipsec :=G F
 \clubpenalty := 10000 % local
 \hskip -\parindent
 \begingroup
 \svsechd
 \endgroup
 \unskip
 \hskip -AFTERSKIP \relax
 %% relax added 14 Jan 91
 ELSE \clubpenalty := \@clubpenalty % local
 \everypar := NULL
 FI
 }
 FI
END
```

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

```
87 \def\@xsect#1{%
88 \tempskipa #1\relax
89 \ifdim \tempskipa>\z@
```

Why not combine \@sect and \@xsect and save doing the same test twice? It is not possible to change this now as these have become hooks!

This \par seems unnecessary.

```
90 \par \nobreak
91 \vskip \tempskipa
92 \afterheading
93 \else
94 \nobreakfalse
95 \global\noskipsectrue
96 \everypar{%
97 \ifnoskipsec
98 \global\noskipsecfalse
99 {\setbox\lastbox}%
100 \clubpenalty\OM
101 \begingroup \svsechd \endgroup
102 \unskip
103 \tempskipa #1\relax
```

```

104 \hskip -\tempskipa
105 \else
106 \clubpenalty \clubpenalty
107 \everypar{}%
108 \fi}%
109 \fi
110 \ignorespaces}

```

(End definition for \@xsect.)

\@seccntformat This command formats the section number including the space following it.

```

111 \def\@seccntformat#1{\csname the#1\endcsname\quad}

```

(End definition for \@seccntformat.)

Pseudocode for the \@sect command *Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more)*:

```

\@sect{INDENT}{BEFRESKIP}{AFTERSKIP}{STYLE}{ARG} ==
BEGIN
IF AFTERSKIP > 0
 THEN \begingroup
 STYLE
 \hangfrom{\hskip INDENT}
 {\interlinepenalty 10000 ARG\par}
 \endgroup
ELSE \svsechd == BEGIN STYLE
 \hskip INDENT
 ARG
 END
FI
\@xsect{AFTERSKIP}
END

```

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

Pseudocode for the \@afterheading command *Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more)*:

```

\@afterheading ==
BEGIN
@nobreak :=G true
\everypar := BEGIN IF @nobreak = T
 THEN @nobreak :=G false
 \clubpenalty := 10000 % local
 IF @afterindent = F
 THEN remove \lastbox
 FI
 ELSE \clubpenalty := \clubpenalty % local
 \everypar := NULL
 FI
END

```

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

```

\@ssect
112 \def\@ssect#1#2#3#4#5{%
113 \@tempskipa #3\relax
114 \ifdim \@tempskipa>\z@
115 \begingroup

```

This { used to be after the argument to `\@hangfrom` but was moved here to allow commands such as `\MakeUppercase` to be used at the end of #4.

```

116 #4{%
117 \@hangfrom{\hskip #1}%
118 \interlinepenalty \OM #5\@par}%
119 \endgroup
120 \else
121 \def\@svsechd{#4{\hskip #1\relax #5}}%
122 \fi
123 \@xsect{#3}

```

(End definition for `\@ssect`.)

```

\if@afterindent
\@afterindenttrue
124 \newif\if@afterindent \@afterindenttrue

```

(End definition for `\if@afterindent` and `\@afterindenttrue`.)

`\@afterheading` This hook is used in setting up custom-built headings in classes.dtx.

```

125 \def\@afterheading{%
126 \nobreaktrue
127 \everypar{%
128 \ifnobreak
129 \nobreakfalse
130 \clubpenalty \OM
131 \if@afterindent \else
132 {\setbox\z@\lastbox}%
133 \fi
134 \else
135 \clubpenalty \clubpenalty
136 \everypar{}%
137 \fi}}

```

(End definition for `\@afterheading`.)

`\@hangfrom` `\@hangfrom{<text>}` : Puts `<text>` in a box, and makes a hanging indentation of the following material up to the first `\par`. Should be used in vertical mode.

```

138 \def\@hangfrom#1{\setbox\@tempboxa\hbox{#1}%
139 \hangindent \wd\@tempboxa\noindent\box\@tempboxa}

```

(End definition for `\@hangfrom`.)

```

\c@secnumdepth
\c@tocdepth
140 \newcount\c@secnumdepth
141 \newcount\c@tocdepth

```

(End definition for `\c@secnumdepth` and `\c@tocdepth`.)

```
\secdef \secdef{\{unstarcmds\}}{\{unstarcmds\}}{\{starcmds\}}
When defining a \chapter or \section command without using \startsection, you
can use \secdef as follows:
```

1. \def\chapter{ ... \secdef {\starcmd} {\unstarcmd} }
  2. \def{\starcmd}[#1]{#2{...}} % Command to define \chapter[...]{...}
  3. \def{\unstarcmd}{#1{...}} % Command to define \chapter\*{...}
- <sup>142</sup> \def\secdef{\#1{\#2{\@ifstar{\#2}{\@dblarg{\#1}}}}}

(End definition for \secdef.)

### 1.2.1 Initializations

```
\sectionmark
\subsectionmark
\subsubsectionmark
\paragraphmark
\ subparagraphmark
143 \let\sectionmark\@gobble
144 \let\subsectionmark\@gobble
145 \let\subsubsectionmark\@gobble
146 \let\paragraphmark\@gobble
147 \let\ subparagraphmark\@gobble
```

(End definition for \sectionmark and others.)

<sup>148</sup> \message{contents,}

## 1.3 Table of Contents etc.

### 1.3.1 Convention

\tf@{foo} = file number for output for table foo. The file is opened only if @filesw = true.

### 1.3.2 Commands

A \l@{type}{\{entry\}}{\{page\}} Macro needs to be defined by document style for making an entry of type *type* in a table of contents, etc. E.g., the document style should define \l@chapter, \l@section, etc.

**Note:** When the \protect command is used in the *entry* or *text* of one of the commands below, it causes the following control sequence to be written on the file without being expanded. The sequence will be expanded when the table of contents entry is processed.

**Surprise:** Inside an \addcontentsline or \addtocontents command argument, the commands: \index, \glossary, and \label are no-ops. This could cause a problem if the user puts an \index or \label into one of the commands he writes, or into the optional ‘short version’ argument of a \section or \caption command.

\starttoc The \starttoc{\ext} command is used to define the commands:  
\tableofcontents, \listoffigures, etc.

For example: \starttoc{lof} is used in \listoffigures. This command reads the .\ext file and sets up to write the new .\ext file.

*Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*

\starttoc{EXT} ==

```

BEGIN
 \begingroup
 \makeatletter
 read file \jobname.EXT
 IF @filesw = true
 THEN open \jobname.EXT as file \tf@EXT
 FI
 @nobreak :=G FALSE %% added 24 May 89
 \endgroup
END

```

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

```

149 \def\@starttoc#1{%
150 \begingroup
151 \makeatletter
152 \cinput{\jobname.#1}%
153 \if@filesw
154 \expandafter\newwrite\curname tf@#1\endcsname
155 \immediate\openout \curname tf@#1\endcsname \jobname.#1\relax
156 \fi
157 \nobreakfalse
158 \endgroup}

```

*(End definition for \@starttoc.)*

\addcontentsline The \addcontentsline{\langle table\rangle}{\langle type\rangle}{\langle entry\rangle} command allows the user to add his/her own entry to a table of contents, etc. The command adds the entry \contentsline{\langle type\rangle}{\langle entry\rangle}{\langle page\rangle}{\{} to the .\langle table\rangle file.

This macro is implemented as an application of \addtocontents. Note that \thepage is not expandable during \protected@write therefore one gets the page number at the time of the \shipout.

```

159 </2ekernel>
160 <2ekernel | latexrelease>
161 <latexrelease>\IncludeInRelease{2020/10/01}%
162 <latexrelease> {\addcontentsline}{fourth argument}%
163 \def\addcontentsline#1#2#3{%

```

We add an empty brace pair at the end of \contentsline so that the number of argument is identical in documents with and without hyperref.

```

164 \addtocontents{#1}{\protect\contentsline{#2}{#3}{\thepage}{\{}%
165 \protected@file@percent}%
166 </2ekernel | latexrelease>
167 <latexrelease>\EndIncludeInRelease
168 <latexrelease>\IncludeInRelease{2018/12/01}%
169 <latexrelease> {\addcontentsline}{Mask line endings}%
170 <latexrelease> \def\addcontentsline#1#2#3{%
171 <latexrelease> \addtocontents{#1}{\protect\contentsline{#2}{#3}{\thepage}}%

```

We add \protected@file@percent at the end which is turned inside \@writefile into a percent character to mask the newline after the closing argument brace.

```

172 <latexrelease> \protected@file@percent}%
173 <latexrelease>\EndIncludeInRelease
174 <latexrelease>\IncludeInRelease{0000/00/00}%
175 <latexrelease> {\addcontentsline}{Mask line endings}%

```

```

176 ⟨latexrelease⟩\def\addcontentsline#1#2#3{%
177 ⟨latexrelease⟩ \addtocontents{#1}{\protect\contentsline{#2}{#3}{\thepage}}}
178 ⟨latexrelease⟩\EndIncludeInRelease
179 ⟨*2ekernel⟩

```

(End definition for `\addcontentsline`.)

`\addtocontents` The `\addtocontents{⟨table⟩}{⟨text⟩}` command adds `⟨text⟩` to the `.⟨table⟩` file, with no page number.

```

180 \long\def\addtocontents#1#2{%
181 \protected@write\@auxout
182 {\let\label\@gobble \let\index\@gobble \let\glossary\@gobble}%
183 {\string\@writefile{#1}{#2}}}

```

(End definition for `\addtocontents`.)

`\contentsline` The `\contentsline{⟨type⟩}{⟨entry⟩}{⟨page⟩}{}` macro produces a `⟨type⟩` entry in a table of contents, etc. It will appear in the `.toc` or other file. For example, The entry for subsection 1.4.3 in the table of contents for example, might be produced by:

```

\contentsline{subsection}
{\makebox{30pt}[r]{1.4.3} Gnats and Gnus}{22}

```

The `\protect` command causes command sequences to be written without expanding them.

```

184 ⟨/2ekernel⟩
185 ⟨*2ekernel | latexrelease⟩
186 ⟨latexrelease⟩\IncludeInRelease{2021/11/15}%
187 ⟨latexrelease⟩ {\contentsline}{Four arguments}%

```

In the toc file `\contentsline` is followed by 4 arguments these days, but only the first 3 are used in the old interface. The fourth was by default empty and only used when `hyperref` was loaded. We now pick up all 4 arguments, save the last one away in `\@contentsline@destination` and then call the old interface. This is done to simplify the interface to `hyperref` and to prepare for future changes.

```

188 \def\contentsline#1#2#3#4{\gdef\@contentsline@destination{#4}%
189 \csname l@#1\endcsname{#2}{#3}}

```

Default definition.

```

190 \let\@contentsline@destination\empty
191 ⟨/2ekernel | latexrelease⟩
192 ⟨latexrelease⟩\EndIncludeInRelease
193 ⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
194 ⟨latexrelease⟩ {\contentsline}{Four arguments}%
195 ⟨latexrelease⟩
196 ⟨latexrelease⟩\def\contentsline#1{\csname l@#1\endcsname}
197 ⟨latexrelease⟩\let\@contentsline@destination\undefined
198 ⟨latexrelease⟩\EndIncludeInRelease
199 ⟨*2ekernel⟩

```

(End definition for `\contentsline`.)

`\@dottedtocline{⟨level⟩}{⟨indent⟩}{⟨numwidth⟩}{⟨title⟩}{⟨page⟩}`: Macro to produce a table of contents line with the following parameters:

**level** If  $\langle level \rangle > \c@tocdepth$ , then no line produced.

**indent** Total indentation from the left margin.

**numwidth** Width of box for number if the  $\langle title \rangle$  has a `\numberline` command. As of 25 Jan 1988, this is also the amount of extra indentation added to second and later lines of a multiple line entry.

**title** Contents of entry.

**page** Page number.

Uses the following parameters, which must be set by the document style. They should be defined with `\def`'s.

**pnumwidth** Width of box in which page number is set.

**tocrmarg** Right margin indentation for all but last line of multiple-line entries.

**dotsep** Separation between dots, in mu units. Should be `\def`'d to a number like 2 or 1.7

#### \@dottedtocline

```
200 {/2ekernel}
201 {*2ekernel | latexrelease}
202 \ifx\latexrelease\relax\else\IncludeInRelease{2018/12/01}\fi%
203 \ifx\latexrelease\relax\else{\@dottedtocline}{\PreventProtrusion}\fi%
204 \def\@dottedtocline#1#2#3#4#5{%
205 \ifnum #1>\c@tocdepth \else
206 \vskip \z@ \relax
207 {\leftskip #2\relax \rightskip \c@tocrmarg \parfillskip -\rightskip
208 \parindent #2\relax\afterindenttrue
209 \interlinepenalty\@M
210 \leavevmode
211 \tempdima #3\relax
212 \advance\leftskip \tempdima \null\nobreak\hspace{-\leftskip
213 \dotsep\relax
214 \leaders\hbox{$\m@th
```

If a document uses fonts other than computer modern, the use of a dot from math can be very disturbing despite the fact that this might be the only place in a document that then uses computer modern. Therefore we surround the dot with an `\hbox` to escape to the surrounding text font.

```
215 \mkern \dotsep mu\hbox{.}\mkern \dotsep
216 mu$\hfill
217 \nobreak
218 \hbox{\normalfont \normalcolor #5}
```

We finish off by preventing any protrusion if that is enabled. If protrusion happens the number may shift to the right and as a result you may end up with an additional dot in the toc line in some situations.

```
219 \kern-\p@\kern\p@}%
220 \par}%
221 \fi}
```

(End definition for \dottedtocline.)

- \noprotrusion This command, if placed directly to the right (or left) of a word, will prevent protrusion of that word into the margin. It is used in the toc entry lines as they shouldn't protrude. It is implemented as to kerns that cancel each other but being there hide the word so that protrusion is not added. Note that a zero kern or an empty box would not work as the protrusion mechanism will skip over those.

222 \DeclareRobustCommand\noprotrusion{\leavevmode\kern-\p@\kern\p@}

(End definition for \noprotrusion.)

```
223 </2ekernel | latexrelease>
224 <latexrelease>\EndIncludeInRelease
225 <latexrelease>\IncludeInRelease{0000/00/00}%
226 <latexrelease> {\@dottedtocline}{Prevent protrusion}%
227 <latexrelease>\def\@dottedtocline#1#2#3#4#5{%
228 <latexrelease> \ifnum #1>\c@tocdepth \else
229 <latexrelease> \vskip \z@ \oplus .2\p@
230 <latexrelease> {\leftskip #2\relax \rightskip \z@ \parfillskip -\rightskip
231 <latexrelease> \parindent #2\relax \afterindenttrue
232 <latexrelease> \interlinepenalty\OM
233 <latexrelease> \leavevmode
234 <latexrelease> \tempdima #3\relax
235 <latexrelease> \advance\leftskip \tempdima \null\nobreak\hskip -\leftskip
236 <latexrelease> {#4}\nobreak
237 <latexrelease> \leaders\hbox{$\m@th
238 <latexrelease> \mkern \z@ \dotsep \mu\hbox{.}\mkern \z@ \dotsep
239 <latexrelease> \mu$}\hfill
240 <latexrelease> \nobreak
241 <latexrelease> \hb@xt@\pnumwidth{\hfil\normalfont \normalcolor #5}%
242 <latexrelease> \par}%
243 <latexrelease> \fi}
244 <latexrelease>
245 <latexrelease>\let\noprotrusion\undefined
246 <latexrelease>\EndIncludeInRelease
247 </2ekernel>
```

**Note:** \nobreak's added 7 Jan 86 to prevent bad line break that left the page number dangling by itself at left edge of a new line.

Changed 25 Jan 88 to use \leftskip instead of \hangindent so leaders of multiple-line contents entries would line up properly.

- \numberline \numberline{\langle number\rangle}: For use in a \contentsline command. It puts \langle number\rangle flush-left in a box of width \tempdima (Before 25 Jan 88 change, it also added \tempdima to the hanging indentation.)

248 \def\numberline#1{\hb@xt@\tempdima{\hfil}}
249 </2ekernel>

(End definition for \numberline.)

# File P

## ltfloat.dtx

### 1 Floats

The different types of floats are identified by a *<type>* name, which is the name of the counter for that kind of float. For example, figures are of type ‘figure’ and tables are of type ‘table’. Each *<type>* has associated a positive *<type number>*, which is a power of two. E.g.,

figures might be have type number 1, tables type number 2, programs type number 4, etc.

The locations where a float can go are specified by a *<placement specifier>*, which is a list of the possible locations, each denoted by a letter as follows:

- h : here — at the current location in the text.
- t : top — at the top of a text page.
- b : bottom — at the bottom of a text page.
- p : page — on a separate float page

In addition, in conjunction with these, you can use ‘!’ which means that the current values of the float positioning parameters are ignored for this float. (Has no effect on ‘p’, float page positioning.) For example, ‘pht’ specifies that the float can appear in any of three locations: page, here or top.

#### 1.1 Floating Environments

```
1 {*2ekernel}
2 \message{floats,}
```

*Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*

Where floats may appear on a page, and how many may appear there are specified by the following float placement parameters. The numbers are named like counters so the user can set them with the ordinary counter-setting commands.

```
\c@topnumber : Number of floats allowed at the top of a column.
\topfraction : Fraction of column that can be devoted to floats.
\c@dbltopnumber, \dbltopfraction
 : Same as above, but for double-column floats.
\c@bottomnumber, \bottomfraction
 : Same as above for bottom of page.
\c@totalnumber : Number of floats allowed in a single column,
 including in-text floats.
{textfraction} : Minimum fraction of column that must contain text.
{floatpagefraction}: Minimum fraction of page that must be taken
 up by float page.
{dblfloatpagefraction}
 : Same as above, for double-column floats.
```

The document style must define the following.

```
\fps@TYPE : The default placement specifier for floats of type
 TYPE.

\ftype@TYPE : The type number for floats of type TYPE.

\ext@TYPE : The file extension indicating the file on which the
 contents list for float type TYPE is stored.
 For example, \ext@figure = 'lof'.

\fnum@TYPE : A macro to generate the figure number for a caption.
 For example, \fnum@TYPE == Figure \thefigure.

\@makecaption{NUM}{TEXT} :
 A macro to make a caption, with NUM the value
 produced by \fnum@... and TEXT the text of the caption.
 It can assume it's in a \parbox of the appropriate width.

\@float{TYPE}[PLACEMENT] : This macro begins a float environment for a
 single-column float of type TYPE with PLACEMENT as the placement
 specifier. The default value of PLACEMENT is defined by
 \fps@TYPE. The environment is ended by \end@float.
 E.g., \figure == \@float{figure}, \endfigure == \end@float.

\@float{TYPE}[PLACEMENT] ==
BEGIN
 if hmode then \@bsphack
 \@floatpenalty := -10002
 else \@floatpenalty := -10003
 fi
 \@capttype ==L TYPE
 \@dblflset
 \@fps ==L PLACEMENT
 \@onellevel@sanitize \@fps
 add default PLACEMENT if at most ! in PLACEMENT ==
\@fpsadddefault
 if inner
 then LaTeX Error: 'Not in outer paragraph mode.'
 \@floatpenalty := 0
 else if \@freelist nonempty
 then \@currbox :=L head of \@freelist
 \@freelist :=G tail of \@freelist
 \count@\currbox :=G 32*\ftype@TYPE +
 bits determined by PLACEMENT
 else \@floatpenalty := 0
 LaTeX Error: 'Too many unprocessed floats'
 fi
 fi
 \@currbox :=G \color@vbox
```

```

\normalcolor
\vbox{
%% 15 Dec 87 -
%% removed \boxmaxdepth :=L 0pt
%% that made box 0 depth because it screwed
%% things up. Instead, added \vskip0pt at end
\hsize = \columnwidth
\@parboxrestore
\@floatboxreset
END

\caption ==
BEGIN
\refstepcounter{@capter}
\@dblarg{@caption{@capter}}
END

```

In following definition, `\par` moved from after `\addcontentsline` to before `\addcontentsline` because the `\write` could cause an extra blank line to be added to the paragraph above the caption. (Change made 12 Jun 87)

```

\@caption{TYPE}[STEXT]{TEXT} ==
BEGIN
\par
\addcontentsline{\ext@TYPE}{TYPE}{\numberline{\theTYPE}{STEXT}}
\begingroup
\@parboxrestore
\normalsize
\@makecaption{\fnum@TYPE}{TEXT}
\par
\endgroup
END

```

`\@dblfloat{TYPE}[PLACEMENT]` : Macro to begin a float environment for a double-column float of type TYPE with PLACEMENT as the placement specifier. The default value of PLACEMENT is 'tp'. The environment is ended by `\end@dblfloat`.  
E.g., `\figure*` == `\@dblfloat{figure}`,  
`\endfigure*` == `\end@dblfloat`.

```

\@dblfloat{TYPE}[PLACEMENT] ==
Identical to \@float{TYPE}[PLACEMENT] except \hsize and \linewidth
are set to \textwidth.
End of historical LATEX 2.09 comments.

```

`\@floatpenalty`  
<sup>3</sup> `\newcount\@floatpenalty`

(End definition for `\@floatpenalty`.)

`\caption` This is set to be an error message outside a float since no `capttype` is defined there; this may need to be changed by some classes.

```

4 \def\caption{%
5 \ifx\@capttype\undefined
6 \@latex@error{\noexpand\caption outside float}\@ehd
7 \expandafter\@gobble
8 \else
9 \refstepcounter\@capttype
10 \expandafter\@firstofone
11 \fi
12 { \@dblarg{\@caption\@capttype} }%
13 }

```

(End definition for `\caption`.)

`\@caption`

```

14 \long\def\@caption#1[#2]#3{%
15 \par
16 \addcontentsline{\csname ext@\#1\endcsname}{\#1}%
17 {\protect\numberline{\csname the#\#1\endcsname}{\ignorespaces #2}}%
18 \begingroup

```

The paragraph setting parameters are normalised at this point, however `\@parboxrestore` resets `\everypar` which is not correct in this context so `\@setminipage` is called if needed.

The float mechanism, like `minipage`, sets the flag `@minipage` true before executing the user-supplied text. Many L<sup>A</sup>T<sub>E</sub>X constructs test for this flag and do not add vertical space when it is true. The intention is that this emulates T<sub>E</sub>X's 'top of page' behaviour. The flag must be set false at the start of the first paragraph. This is achieved by a redefinition of `\everypar`, but the call to `\@parboxrestore` removes that redefinition, so it is re-inserted if needed. If the flag is already false then the `\caption` was not the first entry in the float, and so some other paragraph has already activated the special `\everypar`. In this case no further action is needed.

```

19 \@parboxrestore
20 \if@minipage
21 \@setminipage
22 \fi
23 \normalsize
24 \makecaption{\csname fnum@\#1\endcsname}{\ignorespaces #3}\par
25 \endgroup

```

(End definition for `\@caption`.)

`\@float`

```

\@dblfset 26 \def\@float#1{%
27 \@ifnextchar[%]
28 {\@xflocat{\#1}}%
29 {\edef\reserved@a{\noexpand\@xfloat{\#1}[\csname fps@\#1\endcsname]}%
30 \reserved@a}

```

(End definition for `\@float` and `\@dblfset`.)

`\@dblffloat`

```

31 \def\@dblffloat{%
32 \if@twocolumn\let\reserved@a\@dbflft\else\let\reserved@a\@float\fi
33 \reserved@a

```

(End definition for \dblfloat.)

\fps@dbl Note that all double floats have default fps ‘tp’.

(End definition for \fps@dbl.)

\@setfps This sets the fps, dealing with error conditions by adding the default.

(End definition for \@setfps.)

\@xfloat The first part of this sets the count register that stores all the information about the type and fps of the float.

We assume here that the default specifiers already contain no active characters.

It may be better to store the defaults as numbers, rather than symbol strings.

```
34 </2ekernel>
35 <latexrelease>\IncludeInRelease{2015/01/01}%
36 <latexrelease> {\@xfloat}{Check float options}%
37 <2ekernel | latexrelease>
38 \def\@xfloat #1[#2]{%
39 \@nодокумент
40 \def \@capttype {#1}%
41 \def \@fps {#2}%
42 \@onelv@sanitize \@fps
43 \def \@reserved@b {!}%
44 \ifx \@reserved@b \@fps
45 \@fpsadddefault
46 \else
47 \ifx \@fps \@empty
48 \@fpsadddefault
49 \fi
50 \fi
51 \@ifhmode
52 \@bsphack
53 \@floatpenalty -\@Mii
54 \else
55 \@floatpenalty-\@Miii
56 \fi
57 \@inner
58 \@parmoderr\@floatpenalty\z@
59 \else
60 \@next@\currbox\@freelist
61 {%
62 \@tempcnta \sixt@n
63 \expandafter \@tfor \expandafter \reserved@a
64 \expandafter :\expandafter =\@fps
65 \do
```

Start of changes, use a nested if structure, ending in an error.

```
66 {%
67 \if \@reserved@a h%
68 \@ifodd \@tempcnta
69 \else
70 \advance \@tempcnta \one
71 \fi
```

```

72 \else\if \reserved@a t%
73 \@setfpsbit \tw@%
74 \else\if \reserved@a b%
75 \@setfpsbit 4%
76 \else\if \reserved@a p%
77 \@setfpsbit 8%
78 \else\if \reserved@a !%
79 \ifnum \tempcnta>15
80 \advance\tempcnta -\sixt@@n\relax
81 \fi
82 \else
83 \@latex@error{Unknown float option `\'\reserved@a'}%
84 {Option `\'\reserved@a' ignored and `p' used.}%
85 \@setfpsbit 8%
86 \fi\fi\fi\fi\fi
87 }%

```

End of changes

```

88 \tempcntb \csname ftype@\capttype \endcsname
89 \multiply \tempcntb \xxxii
90 \advance \tempcnta \tempcntb
91 \global \count\currbox \tempcnta
92 }%
93 \fltovf
94 \fi

```

The remainder sets up the box in which the float is typeset, and the typesetting environment to be used. It is essential to have the extra box to avoid the unwanted space that would otherwise often be put at the top of the float.

It ends with a hook; not sure how useful this is but it is needed at present to deal with double-column floats.

```

95 \global \setbox\currbox
96 \color@vbox
97 \normalcolor
98 \vbox \bgroup
99 \hsize\columnwidth
100 \parboxrestore
101 \floatboxreset
102 }%
103 </2ekernel | latexrelease>
104 <latexrelease>\EndIncludeInRelease
105 <latexrelease>\IncludeInRelease{0000/00/00}%
106 <latexrelease> {\@xfloat}{Check float options}%
107 <latexrelease>\def\@xfloat #1[#2]{%
108 <latexrelease> \nодокумент
109 <latexrelease> \def \@capttype {#1}%
110 <latexrelease> \def \@fps {#2}%
111 <latexrelease> \onelevel@sanitize \@fps
112 <latexrelease> \def \reserved@b {!}%
113 <latexrelease> \ifx \reserved@b \@fps
114 <latexrelease> \@fpsadddefault
115 <latexrelease> \else
116 <latexrelease> \ifx \@fps \empty
117 <latexrelease> \@fpsadddefault

```

```

118 <|latexrelease> \fi
119 <|latexrelease> \fi
120 <|latexrelease> \ifhmode
121 <|latexrelease> \@bsphack
122 <|latexrelease> \@floatpenalty -\@Mii
123 <|latexrelease> \else
124 <|latexrelease> \@floatpenalty-\@Miii
125 <|latexrelease> \fi
126 <|latexrelease> \ifinner
127 <|latexrelease> \@parmoderr\@floatpenalty\z@\@next\@currbox\@freelist
128 <|latexrelease> \else
129 <|latexrelease> \@tempcnta \sixt@@n
130 <|latexrelease> \expandafter \@tfor \expandafter \reserved@a
131 <|latexrelease> \expandafter :\expandafter =\@fps
132 <|latexrelease> \do
133 <|latexrelease> \%
134 <|latexrelease> \if \reserved@a h%
135 <|latexrelease> \ifodd \@tempcnta
136 <|latexrelease> \else
137 <|latexrelease> \advance \@tempcnta \one
138 <|latexrelease> \fi
139 <|latexrelease> \if \reserved@a t%
140 <|latexrelease> \@setfpsbit \tw@
141 <|latexrelease> \fi
142 <|latexrelease> \if \reserved@a b%
143 <|latexrelease> \@setfpsbit 4%
144 <|latexrelease> \fi
145 <|latexrelease> \if \reserved@a p%
146 <|latexrelease> \@setfpsbit 8%
147 <|latexrelease> \fi
148 <|latexrelease> \if \reserved@a !%
149 <|latexrelease> \ifnum \@tempcnta>15
150 <|latexrelease> \advance\@tempcnta -\sixt@@n\relax
151 <|latexrelease> \fi
152 <|latexrelease> \fi
153 <|latexrelease> \fi
154 <|latexrelease> \fi
155 <|latexrelease> \fi
156 <|latexrelease> \}%
157 <|latexrelease> \@tempcntb \csname ftype@\@capttype \endcsname
158 <|latexrelease> \multiply \@tempcntb \xxxii
159 <|latexrelease> \advance \@tempcnta \@tempcntb
160 <|latexrelease> \global \count\@currbox \@tempcnta
161 <|latexrelease> \}%
162 <|latexrelease> \@fltovf
163 <|latexrelease> \fi
164 <|latexrelease> \global \setbox\@currbox
165 <|latexrelease> \color@vbox
166 <|latexrelease> \normalcolor
167 <|latexrelease> \vbox \bgroup
168 <|latexrelease> \hsize\columnwidth
169 <|latexrelease> \parboxrestore
170 <|latexrelease> \floatboxreset
171 <|latexrelease> \}%

```

```

172 ⟨latexrelease⟩\EndIncludeInRelease
173 ⟨*2ekernel⟩

(End definition for \@xfloat.)
```

\@floatboxreset The rational for allowing these normally global flags to be set locally here, via \parboxrestore, was stated originally by Donald Arseneau and extended by Chris Rowley. It is because these flags are only set globally to true by section commands, and these should never appear within marginals or floats or, indeed, in any group; and they are only ever set globally to false when they are definitely true.

If anyone is unhappy with this argument then both flags should be treated as in \setnobreak; otherwise this command will be redundant.

```

174 \def \@floatboxreset {%
175 \reset@font
176 \normalsize
177 \@setminipage
178 }
```

(End definition for \@floatboxreset.)

\@setnobreak

```

179 \def \@setnobreak{%
180 \if@nobreak
181 \let\outer@nobreak\@nobreaktrue
182 \else
183 \fi
184 }
```

(End definition for \@setnobreak.)

\@setminipage

```

185 \def \@setminipage{%
186 \ominipagetrue
187 \everypar{\ominipagefalse\everypar{}}
188 }
```

(End definition for \@setminipage.)

\end@float

```

189 \def\end@float{%
190 \endfloatbox
191 \ifnum\@floatpenalty <\z@
```

We make sure that we never exceed \textheight, otherwise float will never get typeset (91/03/15 FMi).

```

192 \largefloatcheck
193 \cons\currlist\currbox
194 \ifnum\@floatpenalty <-\@Mii
195 \penalty -\@Miv
```

Saving and restoring \prevdepth added 26 May 87 to prevent extra vertical space when used in vertical mode.

```

196 \tempdima\prevdepth
197 \vbox{}%
198 \prevdepth\tempdima
```

```

199 \penalty\@floatpenalty
200
201 \else
202 \vadjust{\penalty -\@Miv \vbox{} \penalty\@floatpenalty}\@EspHack
203 \fi
204 \fi
205 }

```

(End definition for \end@float.)

```

\end@dblfloat
205 </2ekernel>
206 <|latexrelease|\IncludeInRelease{2015/01/01}%
207 <|latexrelease| {\end@dblfloat}{float order in 2-column}%
208 <*2ekernel | latexrelease>
209 \def\end@dblfloat{%
210 \if@twocolumn
211 \endfloatbox
212 \ifnum\@floatpenalty <\z@
213 \largefloatcheck

```

Force the depth of two column float boxes.

```

214 \global\dp\@currbox1sp %

```

What follows is essentially \end@float without a starting \endfloatbox.

```

215 \cons\@currlist\@currbox
216 \ifnum\@floatpenalty <-\@Mi
217 \penalty -\@Miv
218 \tempdima\prevdepth
219 \vbox{}%
220 \prevdepth\tempdima
221 \penalty\@floatpenalty
222 \else
223 \vadjust{\penalty -\@Miv \vbox{} \penalty\@floatpenalty}\@EspHack
224 \fi
225
226 \fi
227 \else
228 \end@float
229 \fi
230 }%
231 </2ekernel | latexrelease>
232 <|latexrelease|\EndIncludeInRelease
233 <|latexrelease|\IncludeInRelease{0000/00/00}%
234 <|latexrelease|\def\end@dblfloat{%
235 <|latexrelease|\if@twocolumn
236 <|latexrelease| \endfloatbox
237 <|latexrelease| \ifnum\@floatpenalty <\z@

```

We make sure that we never exceed \textheight, otherwise float will never get typeset (91/03/15 FMI).

```

238 <|latexrelease| \largefloatcheck
239 <|latexrelease| \cons\@dbldeferlist\@currbox
240 <|latexrelease| \fi

```

RmS 92/03/18 changed \c@esphack to \c@Ephack.

```
241 〈\textrlease〉 \ifnum \c@floatpenalty =-\c@Mii \c@Ephack\fi
242 〈\textrlease〉\else
243 〈\textrlease〉 \end\c@float
244 〈\textrlease〉\fi
245 〈\textrlease〉} %
246 〈\textrlease〉\EndIncludeInRelease
247 {*2ekernel} }
```

(End definition for \enddblfloat.)

\c@endfloatbox This macro is not intended to be a hook; it is designed to help maintain the integrity of this code, which is used twice and, as can be seen, is subject to frequent changes.

```
248 \def \c@endfloatbox{ %
249 \par\vskip\z@skip %% \par\vskip\z@ added 15 Dec 87
250 \c@minipagefalse
251 \outer\nobreak
252 \egroup %% end of vbox
253 \color@endbox
254 }
```

(End definition for \c@endfloatbox.)

\outer\nobreak

```
255 \let\outer\nobreak\c@empty
```

(End definition for \outer\nobreak.)

\c@largefloatcheck This calculates by how much a float is oversize for the page and prints this in a warning message.

```
256 \def \c@largefloatcheck{ %
257 \ifdim \ht\c@currbox>\textheight
258 \c@tempdima -\textheight
259 \advance \c@tempdima \ht\c@currbox
260 \c@latex@warning {Float too large for page by \the\c@tempdima} %
261 \ht\c@currbox \textheight
262 \fi
263 }
```

(End definition for \c@largefloatcheck.)

\c@dbfltn  
\c@dblfloat

```
264 \def\c@dbfltn{\c@ifnextchar[{\c@dblfloat[#1]}{\c@dblfloat[#1][tp]}}
265 \def\c@dblfloat[#2]{%
266 \c@xfloat[#1][#2]\hsize\textwidth\linewidth\textwidth}
```

(End definition for \c@dbfltn and \c@dblfloat.)

Moved to ltoutput 93/12/16

```
267 \%newcount\c@topnumber
268 \%newcount\c@dbltopnumber
269 \%newcount\c@bottomnumber
270 \%newcount\c@totalnumber
```

|                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \@floatplacement    | An analysis of \@floatplacement:<br>This should be called whenever \@colht has been set.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|                     | <pre> 271 \def\@floatplacement{\global\@topnum\c@topnumber 272   % Textpage bit, global: 273   \global\@toproom \topfraction\@colht 274   \global\@botnum \c@bottomnumber 275   \global\@botroom \bottomfraction\@colht 276   \global\@colnum \c@totalnumber 277   % Floatpage bit, local: 278   \@fpmin \floatpagefraction\@colht} 279 </pre> <p>(End definition for \@floatplacement.)</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| \@dblfloatplacement | This should be called only within a group. Now changed to provide extra checks in \@addtoblcol, needed when processing a BANG float.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|                     | <pre> 280 &lt;texreleas&gt;\IncludeInRelease{2015/01/01}% 281 &lt;texreleas&gt;      {\@dblfloatplacement}{float order in 2-column}% 282 {*2ekernel   texreleas&gt;} </pre> <p>When making two column float area, look for floats with 1sp depth.</p> <pre> 283 \def\@dblfloatplacement{\global\@dbltopnum\c@dbltopnumber 284   \global\@dbltoproom \dbltopfraction\@colht 285   \@textmin \@colht 286   \advance \@textmin -\@dbltoproom 287   \@fpmin \dblfloatpagefraction\textheight 288   \@fptop \@dblfpptop 289   \@fpsep \@dblfpsep 290   \@fpbot \@dblfpbot </pre> <p>\f@depth is used in \@testwrongwidth to look for either column or dbl-column floats. A value of 1sp signals the latter. Because of this setting here, \@dblfloatplacement needs to be called inside a group which is a questionable design.</p> <pre> 291   \def\f@depth{1sp}% 292 </pre> <p>Textpage bit: global, but need not be.</p> <pre> 297 &lt;texreleas&gt; \global\@dbltopnum\c@dbltopnumber 298 &lt;texreleas&gt; \global\@dbltoproom \dbltopfraction\@colht </pre> <p>This new bit uses \@textmin to locally store the amount of extra room in the column.</p> <pre> 299 &lt;texreleas&gt; \@textmin \@colht 300 &lt;texreleas&gt; \advance \@textmin -\@dbltoproom </pre> <p>Floatpage bit: must be local.</p> <pre> 301 &lt;texreleas&gt; \@fpmin \dblfloatpagefraction\textheight 302 &lt;texreleas&gt; \@fptop \@dblfpptop 303 &lt;texreleas&gt; \@fpsep \@dblfpsep 304 &lt;texreleas&gt; \@fpbot \@dblfpbot 305 &lt;texreleas&gt;}% 306 &lt;texreleas&gt;\EndIncludeInRelease 307 </pre> |

(End definition for \dblfloatplacement.)

Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):

MARGINAL NOTES:

Marginal notes use the same mechanism as floats to communicate with the \output routine. Marginal notes are distinguished from floats by having a negative placement specification. The command \marginpar [LTEXT]{RTEXT} generates a marginal note in a parbox, using LTEXT if it's on the left and RTEXT if it's on the right. (Default is RTEXT = LTEXT.) It uses the following parameters.

```
\marginparwidth : Width of marginal notes.
\marginparsep : Distance between marginal note and text.
 the page layout to determine how to move the marginal
 note into the margin. E.g., \leftmarginskip ==
 \hskip -\marginparwidth \hskip -\marginparsep .
\marginparpush : Minimum vertical separation between \marginpar's
```

Marginal notes are normally put on the outside of the page if @mparswitch = true, and on the right if @mparswitch = false. The command \reversemarginpar reverses the side where they are put. \normalmarginpar undoes \reversemarginpar. These commands have no effect for two-column output.

SURPRISE: if two marginal notes appear on the same line of text, then the second one could appear on the next page, in a funny position.

```
\marginpar [LTEXT]{RTEXT} ==
BEGIN
 if hmode then \bsphack
 \floatpenalty := -10002
 else \floatpenalty := -10003
 fi
 if inner
 then LaTeX Error: 'Not in outer paragraph mode.'
 \floatpenalty := 0
 else if \freelist has two elements:
 then get \marbox, \currbox from \freelist
 \count\marbox := G -1
 else \floatpenalty := 0
 LaTeX Error: 'Too many unprocessed floats'
 \currbox, \marbox := \tempboxa %%use \def
 fi
 fi
 if optional argument
 then %% \xmpar ==
 \savemarbox\marbox{LTEXT}
 \savemarbox\currbox{RTEXT}
```

```

else %% \@ympar ==
 \@savemarbox\@marbox{RTEXT}
 \box\@currbox :=G \box\@marbox
fi
\@xympar
END

\reversemarginpar == BEGIN \@mparbottom :=G 0
 @reversemargin :=G true
END

\normalmarginpar == BEGIN \@mparbottom :=G 0
 @reversemargin :=G false
END

```

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

```

\marginpar
308 \def\marginpar{%
309 \ifhmode
310 \bsphack
311 \floatpenalty -\Mii
312 \else
313 \floatpenalty-\Miii
314 \fi
315 \ifinner
316 \parmoderr
317 \floatpenalty\z@
318 \else
319 \next\@currbox\@freelist{}{%
320 \next\@marbox\@freelist{\global\count\@marbox\m@ne}%
321 {\floatpenalty\z@
322 \f@tovf\def\@currbox{\tempboxa}\def\@marbox{\tempboxa}}%
323 \fi
324 \ifnextchar [\@xmpar\@ympar}

```

*(End definition for \marginpar.)*

```

\@xmpar
325 \long\def\@xmpar[#1]{%
326 \@savemarbox\@marbox{#1}%
327 \@savemarbox\@currbox{#2}%
328 \@xympar}

```

*(End definition for \@xmpar.)*

```

\@ympar
329 \long\def\@ympar#1{%
330 \@savemarbox\@marbox{#1}%
331 \global\setbox\@currbox\copy\@marbox
332 \@xympar}

```

*(End definition for \@ympar.)*

```

\@savemarbox
 333 </2ekernel>
 334 <*2ekernel | latexrelease>
 335 <latexrelease>\IncludeInRelease{2021/06/01}%
 336 <latexrelease> {\@savemarbox}{Explicit par for marginpar}%
 337 \long\def \@savemarbox #1#2{%
 338 \global\setbox #1%
 339 \color@vbox
 340 \vtop{%
 341 \hsize\marginparwidth
 342 \parboxrestore
 343 \marginparreset
 344 #2\par
 345 \minipagetrue
 346 \outer@nobreak
 347 }%
 348 \color@endbox
 349 }
 350 </2ekernel | latexrelease>
 351 <latexrelease>\EndIncludeInRelease
 352 <latexrelease>\IncludeInRelease{0000/00/00}%
 353 <latexrelease> {\@savemarbox}{Explicit par for marginpar}%
 354 <latexrelease>
 355 <latexrelease>\long\def \@savemarbox #1#2{%
 356 \global\setbox #1%
 357 \color@vbox
 358 \vtop{%
 359 \hsize\marginparwidth
 360 \parboxrestore
 361 \marginparreset
 362 #2%
 363 \minipagetrue
 364 \outer@nobreak
 365 }%
 366 \color@endbox
 367 <latexrelease>}
 368 <latexrelease>\EndIncludeInRelease
 369 <*2ekernel>

```

(End definition for `\@savemarbox`.)

`\@marginparreset` The rational for allowing these normally global flags to be set locally here, via `\parboxrestore` was stated originally by Donald Arseneau and extended by Chris Rowley. It is because these flags are only set globally to true by section commands, and these should never appear within marginals or floats or, indeed, in any group; and they are only ever set globally to false when they are definitely true.

If anyone is unhappy with this argument then both flags should be treated as in `\set@nobreak`; otherwise this command will be redundant.

```

370 \def \@marginparreset {%
371 \reset@font
372 \normalsize
373 \% \let\if@nobreak\iffalse
374 \% \let\if@noskipsec\iffalse

```

```

375 % \@setnobreak
376 \@setminipage
377 }

(End definition for \@marginparreset.)
```

### \@xympar

Setting the box here is done only because the code uses `\end@float`; it will be empty and gets discarded.

```

378 \def \@xympar{%
379 \ifnum\@floatpenalty <\z@\@cons\@currlist\@marbox\fi
380 \setbox\@tempboxa
381 \color@vbox
382 \vbox \bgroup
383 \end@float
384 \ignorespacesfalse
385 \esphack
386 }
```

(End definition for `\@xympar`.)

### \reversemarginpar

```

\reversemarginpar
\normalmarginpar
387 \def\reversemarginpar{\global\@mparbottom\z@\@reversemargintrue}
388 \def\normalmarginpar{\global\@mparbottom\z@\@reversemarginfalse}
```

(End definition for `\reversemarginpar` and `\normalmarginpar`.)

```
389 \message{footnotes,}
```

## 1.2 Footnotes

*Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*

`\footnote{NOTE}` : User command to insert a footnote.

`\footnote[NUM]{NOTE}`: User command to insert a footnote numbered *NUM*, where *NUM* is a number – 1, 2, etc. For example, if footnotes are numbered \*, \*\*, etc. within pages, then `\footnote[2]{...}` produces footnote \*\*. This command does not step the footnote counter.

`\footnotemark[NUM]` : Command to produce just the footnote mark in the text, but no footnote. With no argument, it steps the footnote counter before generating the mark.

`\footnotetext[NUM]{TEXT}` : Command to produce the footnote but no mark. `\footnote` is equivalent to `\footnotemark \footnotetext`.

As in PLAIN, footnotes use `\insert\footins`, and the following parameters:

**\footnotesize** : Size-changing command for footnotes.

**\footnotesep** : The height of a strut placed at the beginning of every footnote.

**\skip\footins** : Space between main text and footnotes. The rule separating footnotes from text occurs in this space. This space lies above the strut of height **\footnotesep** which is at the beginning of the first footnote.

**\footnoterule** : Macro to draw the rule separating footnotes from text. It is executed right after a **\vspace** of **\skip\footins**. It should take zero vertical space—i.e., it should skip to a negative value to compensate for any positive space it occupies. (See PLAIN.TEX.)

**\interfootnotelinepenalty** : Interline penalty for footnotes.

**\thefootnote** : In usual LaTeX style, produces the footnote number. If footnotes are to be numbered within pages, then the document style file must include an **\@addtoreset** command to cause the footnote counter to be reset when the page counter is stepped. This is not a good idea, though, because the counter will not always be reset in time to ensure that the first footnote on a page is footnote number one.

**\@thefnmark** : Holds the current footnote's mark—e.g., **\dag** or '1' or 'a'.

**\@mpfnnumber** : A macro that generates the numbers for **\footnote** and **\footnotemark** commands. It == **\thefootnote** outside a **minipage** environment, but can be changed inside to generate numbers for **\footnote**'s.

**\@makefnmark** : A macro to generate the footnote marker from **\@thefnmark**. The default definition was **\hbox{\$^{\@thefnmark}\$}**.

This is now replaced by  
 $\text{\textsuperscript}{\@thefnmark}$

**\@makefntext{NOTE}** :

Must produce the actual footnote, using **\@thefnmark** as the mark of the footnote and NOTE as the text. It is called when effectively inside a **\parbox**, with **\hsize = \columnwidth**. For example, it might be as simple as  
 $\$^{\@thefnmark}\$ \text{ NOTE}$

In a minipage environment, `\footnote` and `\footnotetext` are redefined so that

- (a) they use the counter `mpfootnote`
  - (b) the footnotes they produce go at the bottom of the minipage.
- The switch is accomplished by letting `\@mpfn == footnote` or `mpfootnote` and `\@thempfn == \thefootnote` or `\thempfootnote`, and by redefining `\@footnotetext` to be `\@mpfootnotetext` in the minipage.

```
\footnote{NOTE} ==
BEGIN
 \stepcounter{\@mpfn}
 begingroup
 \protect == \noexpand
 \@thefnmark :=G eval (\thempfn)
 endgroup
 \@footnotemark
 \@footnotetext{NOTE}
END

\footnote[NUM]{NOTE} ==
BEGIN
 begingroup
 \protect == \noexpand
 counter \@mpfn :=L NUM
 \@thefnmark :=G eval (\thempfn)
 endgroup
 \@footnotemark
 \@footnotetext{NOTE}
END

\footnotemark ==
BEGIN \stepcounter{footnote}
 begingroup
 \protect == \noexpand
 \@thefnmark :=G eval(\thefootnote)
 endgroup
 \@footnotemark
END

\footnotemark[NUM] ==
BEGIN
 begingroup
 footnote counter :=L NUM
 \protect == \noexpand
 \@thefnmark :=G eval(\thefootnote)
 endgroup
 \@footnotemark
END
```

```

\@footnotemark ==
BEGIN
 \leavevmode
 IF hmode THEN \c@sf := \the\spacefactor FI
 \c@makefnmark % put number in main text
 IF hmode THEN \spacefactor := \c@sf FI
END

\footnotetext ==
BEGIN begingroup \protect == \noexpand
 \c@thefnmark :=G eval (\thempfn)
endgroup
\@footnotetext
END

\footnotetext[NUM] ==
BEGIN begingroup counter \c@mpfn :=L NUM
 \protect == \noexpand
 \c@thefnmark :=G eval (\thempfn)
endgroup
\@footnotetext
END

```

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

**\footins** L<sup>A</sup>T<sub>E</sub>X does use the same insert for footnotes as PLAIN.

390 \newinsert\footins

L<sup>A</sup>T<sub>E</sub>X leaves these initializations for the \footins insert.

391 \skip\footins=\bigskipamount % space added when footnote is present
392 \count\footins=1000 % footnote magnification factor (1 to 1)
393 \dimen\footins=8in % maximum footnotes per page

(*End definition for \footins.*)

**\footnoterule** L<sup>A</sup>T<sub>E</sub>X keeps PLAIN T<sub>E</sub>X's \footnoterule as the default.

394 \def\footnoterule{\kern-3\p@

395 \hrule \width 2in \kern 2.6\p@} % the \hrule is .4pt high

(*End definition for \footnoterule.*)

**\thefootnote**

396 \c@definecounter{footnote}

397 \def\thefootnote{\c@arabic\c@footnote}

(*End definition for \thefootnote.*)

**\thempfootnote** The default display for the footnote counter in minipages is to use italic letters. We use \itshape not \textit as the latter would add an italic correction.

398 \c@definecounter{mpfootnote}

399 \def\thempfootnote{\itshape\c@alph\c@mpfootnote}}

(*End definition for \thempfootnote.*)

\@makefnmark Default definition.

```

400 \%\\def\\@makefnmark{\\hbox{$^{\\@thefnmark}\\m@th$}}
401 \\def\\@makefnmark{\\hbox{\\\normalfont\\@thefnmark}}

```

(End definition for \@makefnmark.)

\textsuperscript This command provides superscript characters in the current text font. It's implementation might change!!!

```

402 \\DeclareRobustCommand*\\textsuperscript[1]{%
403 \\@textsuperscript{\\selectfont#1}}

```

(End definition for \textsuperscript.)

\@textsuperscript This command should not be used directly, but may be used to define other commands \textsuperscript, \@makefnmark. #1 should always start with a font selection command, to activate the font size switch.

```

404 \\begin{document}
405 \\begin{2ekernel | latexrelease}
406 \\latexrelease\\IncludeInRelease{2020/10/01}%
407 \\latexrelease \\{@textsuperscript}{superscript baseline}%
408 \\def\\@textsuperscript#1{%
409 \\m@th\\ensuremath{^\\{\\mbox{\\footnotesize\\selectfont#1}\\}}}
410 \\end{2ekernel | latexrelease}
411 \\latexrelease\\EndIncludeInRelease
412 \\latexrelease\\IncludeInRelease{0000/00/00}%
413 \\latexrelease \\{@textsuperscript}{superscript baseline}%
414 \\latexrelease
415 \\latexrelease\\def\\@textsuperscript#1{%
416 \\latexrelease \\m@th\\ensuremath{^\\{\\mbox{\\footnotesize\\selectfont#1}\\}}}
417 \\latexrelease\\EndIncludeInRelease
418 \\end{document}

```

(End definition for \@textsuperscript.)

\textsubscript

```

419 \\begin{document}
420 \\begin{2ekernel | latexrelease}
421 \\latexrelease \\{@textsubscript}{\\textsubscript}%
422 \\end{document}
423 \\DeclareRobustCommand*\\textsubscript[1]{%
424 \\@textsubscript{\\selectfont#1}}

```

```

425 \\end{document}
426 \\latexrelease\\EndIncludeInRelease
427 \\latexrelease\\IncludeInRelease{0000/00/00}%
428 \\latexrelease \\{@textsubscript}{\\textsubscript}%
429 \\latexrelease\\let\\textsubscript@undefined
430 \\latexrelease\\EndIncludeInRelease
431 \\end{document}

```

(End definition for \textsubscript.)

```

\@textsubscript
 432 </2ekernel>
 433 {*2ekernel | latexrelease}
 434 <latexrelease>\IncludeInRelease{2020/10/01}%
 435 <latexrelease> {\@textsubscript}{subscript baseline}%
 436 \def\@textsubscript#1{%
 437 {\m@th\ensuremath{_f\mbox{\scriptsize\sffamily\sf@size\sf@size#1}}}}
 438 </2ekernel | latexrelease>
 439 <latexrelease>\EndIncludeInRelease
 440 <latexrelease>\IncludeInRelease{2015/01/01}%
 441 <latexrelease> {\@textsubscript}{subscript baseline}%
 442 <latexrelease>
 443 <latexrelease>\def\@textsubscript#1{%
 444 {\m@th\ensuremath{_f\mbox{\scriptsize\sffamily\sf@size\z@#1}}}}
 445 <latexrelease>\EndIncludeInRelease
 446 <latexrelease>\IncludeInRelease{0000/00/00}%
 447 <latexrelease> {\@textsubscript}{subscript baseline}%
 448 <latexrelease>\let\@textsubscript\undefined
 449 <latexrelease>\EndIncludeInRelease
 450 {*2ekernel}

(End definition for \@textsubscript.)

\footnotesep
 451 \newdimen\footnotesep

(End definition for \footnotesep.)

\footnote
 452 \def\footnote{\@ifnextchar[\@xfootnote{\stepcounter\@mpfn
 453 \protected@xdef\@thefnmark{\thempfn}%
 454 \@footnotemark\@footnotetext}}
 455 <mpfn>\@footnotemark\@footnotetext

(End definition for \footnote.)

\@xfootnote
 455 \def\@xfootnote[#1]{%
 456 \begingroup
 457 \csname c@\@mpfn\endcsname #1\relax
 458 \unrestored@protected@xdef\@thefnmark{\thempfn}%
 459 \endgroup
 460 \@footnotemark\@footnotetext}
 461 </2ekernel>
 462 {*2ekernel | latexrelease}
 463 <latexrelease>\IncludeInRelease{2021/11/15}%
 464 <latexrelease> {\@footnotetext}{footnotetext tagging}%
 465 \long\def\@footnotetext#1{\insert\footins{%
 466 \reset@font\footnotesize
 467 \interlinepenalty\interfootnotelinepenalty
 468 \splittopskip\footnotesep

```

```

469 \splitmaxdepth \dp\strutbox \floatingpenalty \@MM
470 \hsize\columnwidth \parboxrestore
471 \def\@currentcounter{footnote}%
472 \protected@edef\@currentlabel{%
473 \csname p@footnote\endcsname\@thefnmark
474 }%
475 \color@begingroup
476 \makefntext{%
477 \rule{z@\footnotesep}{\ignorespaces#1\finalstrut\strutbox}%
478 \par
479 \color@endgroup}%
480 {/2ekernel | latexrelease}
481 \end{IncludeInRelease}

482 \begin{IncludeInRelease}[2021/06/01]
483 \begin{latexrelease}
484 {\@footnotetext}{footnotetext tagging}%
485 \long\def\@footnotetext#1{\insert\footins{%
486 \reset@font\footnotesize
487 \interlinepenalty\interfootnotelinepenalty
488 \splittopskip\footnotesep
489 \splitmaxdepth \dp\strutbox \floatingpenalty \@MM
490 \hsize\columnwidth \parboxrestore
491 \protected@edef\@currentlabel{%
492 \csname p@footnote\endcsname\@thefnmark
493 }%
494 \color@begingroup
495 \makefntext{%
496 \rule{z@\footnotesep}{\ignorespaces#1\finalstrut\strutbox}%
497 \par
498 \color@endgroup}%
499 \end{IncludeInRelease}
500 \begin{IncludeInRelease}[0000/00/00]
501 \begin{latexrelease}
502 {\@footnotetext}{footnotetext tagging}%
503 \long\def\@footnotetext#1{\insert\footins{%
504 \reset@font\footnotesize
505 \interlinepenalty\interfootnotelinepenalty
506 \splittopskip\footnotesep
507 \splitmaxdepth \dp\strutbox \floatingpenalty \@MM
508 \hsize\columnwidth \parboxrestore
509 \protected@edef\@currentlabel{%
510 \csname p@footnote\endcsname\@thefnmark
511 }%
512 \color@begingroup
513 \makefntext{%
514 \rule{z@\footnotesep}{\ignorespaces#1\finalstrut\strutbox}%
515 \color@endgroup}%
516 \end{IncludeInRelease}
517 {*2ekernel}

```

(End definition for \@footnotetext.)

\footnotemark

```

518 \def\footnotemark{%
519 \ifnextchar[\@xfootnotemark
520 {\stepcounter{footnote}%
521 \protected@xdef\@thefnmark{\thefootnote}%
522 \@footnotemark}%

```

(End definition for `\footnotemark`.)

`\@xfootnotemark`

```

523 \def\@xfootnotemark[#1]{%
524 \begingroup
525 \c@footnote #1\relax
526 \unrestored@protected@xdef\@thefnmark{\thefootnote}%
527 \endgroup
528 \@footnotemark}%

```

(End definition for `\@xfootnotemark`.)

`\@footnotemark`

```

529 \def\@footnotemark{%
530 \leavevmode
531 \ifhmode\edef\@x@sf{\the\spacefactor}\nobreak\fi
532 \makefnmark
533 \ifhmode\spacefactor\@x@sf\fi
534 \relax}%

```

(End definition for `\@footnotemark`.)

`\footnotetext`

```

535 \def\footnotetext{%
536 \ifnextchar [\@xfootnotenext
537 {\protected@xdef\@thefnmark{\thempfn}%
538 \@footnotetext}}%

```

(End definition for `\footnotetext`.)

`\@xfootnotenext`

```

539 \def\@xfootnotenext[#1]{%
540 \begingroup
541 \csname c@\@mpfn\endcsname #1\relax
542 \unrestored@protected@xdef\@thefnmark{\thempfn}%
543 \endgroup
544 \@footnotetext}%

```

(End definition for `\@xfootnotenext`.)

`\thempfn`

```

545 \def\@mpfn{footnote}
546 \def\thempfn{\thefootnote}%

```

(End definition for `\thempfn` and `\@mpfn`.)

**\footref** This command generates a footnote mark. The value is produced by referencing a `\label` placed into a `\footnote` elsewhere (can be one in the main galley or in a minipage).

```
547 ⟨/2ekernel⟩
548 ⟨*2ekernel | latexrelease⟩
549 ⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
550 ⟨latexrelease⟩ {\footref}{Add footref}%
551 \def\footref#1{%
552 \begingroup
553 \unrestored@protected@xdef\@thefnmark{\ref{#1}}%
554 \endgroup
555 \footnotemark
556 }
557 ⟨/2ekernel | latexrelease⟩
558 ⟨latexrelease⟩\EndIncludeInRelease
```

We don't remove it when rolling back so that packages offered it in the past do not need to alter their behavior in a rollback situation.

```
559 ⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
560 ⟨latexrelease⟩ {\footref}{Add footref}%
561 ⟨latexrelease⟩
562 ⟨latexrelease⟩ % \let\footref\@undefined
563 ⟨latexrelease⟩
564 ⟨latexrelease⟩\EndIncludeInRelease
565 ⟨*2ekernel⟩
```

(*End definition for \footref.*)

```
566 ⟨/2ekernel⟩
```

# File Q

## ltidxglo.dtx

### 1 Index and Glossary Generation

Index and Glossary commands.

```
\makeindex A preamble command to turn on indexing.
\makeglossary A preamble command to turn on making glossary entries.
 \index Make an index entry for #1.
 \glossary Make a glossary entry for #1.
Historical LATEX 2.09 comments (not necessarily accurate any more):
\makeindex ==
 BEGIN
 \index == BEGIN \@bsphack
 \begingroup
 \protect{X} == \string X\space
 %% added 3 Feb 87 for \index commands
 %% in \footnotes
 re-\catcode special characters
 to 'other'
 \@wrindex
 END

\@wrindex{ITEM} ==
 BEGIN
 write of {\indexentry{ITEM}{page number}}
 \endgroup
 \@esphack
 END

INITIALIZATION:

\index == BEGIN \@bsphack
 \begingroup
 re-\catcode special characters (in case '%' there)
 \@index
 END

\@index{ITEM} == BEGIN \endgroup \@esphack END

Changes made 14 Apr 89 to write \glossaryentry's instead of
\indexentry's on the .glo file.
End of historical LATEX 2.09 comments.
1 {*2ekernel}
2 \message{index,}
```

```

\makeindex

3 \def\makeindex{%
4 \newwrite\@indexfile
5 \immediate\openout\@indexfile=\jobname.idx
6 \def\index{\@bsphack\begingroup
7 \@sanitize
8 \wrindex}\typeout
9 {Writing index file \jobname.idx}%

```

Opening the write channel should be done only once since on some OS multiple opens are forbidden and in any case it is useless. So we turn this into a no-op after use.

```

10 \let\makeindex\empty
11 }
12 \onlypreamble\makeindex

```

(*End definition for \makeindex.*)

```

\@wrindex

13 \def\@wrindex#1{%
14 \protected@write\@indexfile{}{%
15 {\string\indexentry{#1}{\thepage}}%
16 \endgroup
17 \esphack}

```

(*End definition for \@wrindex.*)

```

\index

18 \def\index{\@bsphack\begingroup \@sanitize\@index}

```

(*End definition for \index.*)

```

\@index

19 \def\@index#1{\endgroup\esphack}

```

(*End definition for \@index.*)

```

\makeglossary

20 \def\makeglossary{%
21 \newwrite\@glossaryfile
22 \immediate\openout\@glossaryfile=\jobname.glo
23 \def\glossary{\@bsphack\begingroup
24 \@sanitize
25 \wrglossary}\typeout
26 {Writing glossary file \jobname.glo }%

```

Opening the write channel should be done only once since on some OS multiple opens are forbidden and in any case it is useless. So we turn this into a no-op after use.

```

27 \let\makeglossary\empty
28 }
29 \onlypreamble\makeglossary

```

(*End definition for \makeglossary.*)

```
\@wrglossary
30 \def\@wrglossary#1{%
31 \protected@write\@glossaryfile{}{%
32 {\string\glossaryentry{\#1}{\thepage}}%
33 \endgroup
34 \@esphack}
(End definition for \@wrglossary.)
```

```
\glossary
35 \def\glossary{\@bsphack\begingroup\@sanitize\@index}
(End definition for \glossary.)
36 </2ekernel>
```

# File R

## ltbibl.dtx

### 1 Bibliography Generation

A bibliography is created by the `thebibliography` environment, which generates a title such as “References”, and a list of entries. The BIBTEX program will create a file containing such an environment, which will be read in by the `\bibliography` command. With BIBTEX, the following commands will be used.

`\bibliography{<file1,<file2, ...,<filen>}` : specifies the bibdata files. Writes a `\bibdata` entry on the `.aux` file and tries to read in `mainfile.bbl`.

`\bibliographystyle{<style>}` : Writes a `\bibstyle` entry on the `.aux` file.

The `thebibliography` environment is a list environment. To save the use of an extra counter, it should use `enumiv` as the item counter. Instead of using `\item`, items in the bibliography are produced by the following commands:

`\bibitem{<name>}` : Produces a numbered entry cited as `<name>`.

`\bibitem[<label>]{<name>}` : Produces an entry labeled by `<Label>` and cited by `<name>`.

The former is used for bibliographies with citations like [1], [2], etc.; the latter is used for citations like [Knuth82].

The document class must define the `thebibliography` environment. This environment has a single argument, which is the widest bibliography label— e.g., if the [Knuth67] is the widest entry, then this argument will be Knuth67. The `\thebibliography` command must begin a list environment, which the `\endthebibliography` command ends.

`\cite` Entries are cited by the command `\cite{<name>}`.

`\nocite{<citations>}` puts information on the `.aux` file that causes BIBTEX to include the `{<citations>}` list in the bibliography, but puts nothing in the text.

`\nocite{*}` is special: it tells BIBTEX to put the whole of a collection of references into the bibliography.

1 `(*2ekernel)`  
2 `\message{bibliography,}`

*Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*

PARAMETERS

`\@cite` : A macro such that `\@cite{LABEL1,LABEL2}{NOTE}` produces the output for a `\cite[NOTE]{FOO1,FOO2}` command, where entry FOOi is defined by `\bibitem[LABELi]{FOOi}`.

The switch @tempswa is true if the optional NOTE argument is present.

The default definition is :

```
\@cite{LABELS}{NOTE} ==
BEGIN [LABELS
 IF @tempswa = T THEN , NOTE FI
]
END
```

`\@biblabel` : A macro to produce the label in the bibliography entry. For `\bibitem[LABEL]{NAME}`, the label is

generated by `\@biblabel{LABEL}`. It has the default definition `\@biblabel{LABEL} -> [LABEL]`.

## CONVENTION

`\b@FOO` : The name or number of the reference created by `\cite{FOO}`  
E.g., if `\cite{FOO} -> [17]`, then `\b@FOO -> 17`.

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

```
\bibitem
 \def\bibitem{\@ifnextchar[\@lbibitem\@bibitem}
(End definition for \bibitem.)

\@lbibitem
 \def@\lbibitem[#1]{\item[\@biblabel{#1}\hfill]\if@filesw
 {\let\protect\noexpand
 \immediate
 \write\auxout{\string\bibcite{#2}{#1}}}\fi\ignorespaces}
(End definition for \@lbibitem.)

\@bibitem
 \def@\bibitem#1{\item\if@filesw \immediate\write\auxout
 {\string\bibcite{#1}{\the\value{\listctr}}}\fi\ignorespaces}
(End definition for \@bibitem.)

\bibcite
 \def\bibcite{\@newl@bel b}
(End definition for \bibcite.)

\citation
 \let\citation\@gobble
(End definition for \citation.)

\cite
 </2ekernel>
 <*2ekernel | latexrelease>
 <latexrelease>\IncludeInRelease{2022/06/01}%
 <latexrelease> {\cite}{check for blank}%
 \DeclareRobustCommand\cite{%
 \@ifnextchar [{\@tempswattrue\@citex@checkblank}{\@tempswafalse\@citex@checkblank[]}}}
```

Due to the way `\for` as used in `\@citex` behaves an empty argument to `\cite` did not produce any warning for a missing citation. So we now inject a command before calling `\@citex` that does the checking for us. It is not done in `\@citex` directly, because that command is altered by a number of packages/classes and this way it is more likely that the check survives.

```
 \def\@citex@checkblank[#1]{%
 \IfBlankTF {#2}{%
 {\@citex[#1]{\space}}%
 {\@citex[#1]{#2}}%
 }%
 }%
</2ekernel | latexrelease>
```

```

24 ⟨latexrelease⟩\EndIncludeInRelease
25 ⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
26 ⟨latexrelease⟩ {\cite}{check for blank}%
27 ⟨latexrelease⟩
28 ⟨latexrelease⟩\DeclareRobustCommand\cite{%
29 ⟨latexrelease⟩ \@ifnextchar [{\@tempswatrue\@citex}{\@tempswafalse\@citex[]}}%
30 ⟨latexrelease⟩\let\@citex@checkblank\@undefined
31 ⟨latexrelease⟩
32 ⟨latexrelease⟩\EndIncludeInRelease
33 {*2ekernel}

```

(*End definition for \cite.*)

**\@citex** \penalty\@m added to definition of \@citex to allow a line break after the ‘,’ in citations like [Jones80,Smith77] (Added 23 Oct 86)  
space added after the ‘,’ (21 Nov 87)

```

34 \def\@citex[#1]{\leavevmode
35 \let\@citema\@empty
36 \@cite{\@for\@citemb:=#2\do
37 {\@citema\def\@citema{,\penalty\@m\ }%
38 \edef\@citemb{\expandafter\@firstofone\@citemb\@empty}%
39 \if@filesw\immediate\write\@auxout{\string\citation{\@citemb}}\fi

```

Using \hbox instead of \mbox is fine because of the \leavevmode above. In fact the use of a box around the citation contents is more than questionable in my view (FMi), but within 2e I have to keep that for compatibility reasons as it would probably change too many existing documents. Its main reason is to avoid hyphenation of labels such as [FOOB89] into [FOO- B89] so in certain styles it makes sense; but, for example, in author year citations it becomes more than questionable.

So Chris added yet another hook here, as suggested by, at least, Donald Arseneau. Note that this one is inside the first argument of the \@cite hook. This decouples the top-level typesetting of the citation from the details of the other business conducted here. All this really needs a complete rethink to get the right modularity.

```

40 \ifundefined{b@\@citemb}{\hbox{\reset@font\bfseries ?}}%
41 \G@refundefinedtrue
42 \@latex@warning
43 {Citation ‘\@citemb’ on page \thepage \space undefined}%
44 {\@cite@ofmt{\csname b@\@citemb\endcsname}}}\#1}}

```

(*End definition for \@citex.*)

```

\bibdata
\bibstyle 45 \let\bibdata=\@gobble
46 \let\bibstyle=\@gobble

```

(*End definition for \bibdata and \bibstyle.*)

\bibliography

```

47 \def\bibliography#1{%
48 \if@filesw
49 \immediate\write\@auxout{\string\bibdata{\zap@space#1 \@empty} }%
50 \fi
51 \@input{\jobname.bbl}

```

(End definition for \bibliography.)

```
\bibliographystyle
52 \def\bibliographystyle#1{%
53 \ifx\@begindocumenthook\@undefined\else
54 \expandafter\AtBeginDocument
55 \fi
56 {\if@filesw
57 \immediate\write\auxout{\string\bibstyle{#1}}%
58 \fi}}
```

(End definition for \bibliographystyle.)

\nocite (Added 14 Jun 85)

This puts information on the .aux file that causes BIBTEX to include the citation list in the bibliography, but puts nothing in the text.

RmS 93/08/06: Made loop for \nocite like that for \@citet, to get rid of leading spaces.

```
59 </2ekernel>
60 <*2ekernel | latexrelease>
61 <| latexrelease>\IncludeInRelease{2021/06/01}%
62 <| latexrelease> {\nocite}{Allow nocite in preamble}%
63 \def\nocite#1{\@bsphack
```

With the implementation designed already in LATEX 2.09 the \nocite command will not work before \begin{document} since it tries to write to the .aux file which is not open before that point. As a result the “reference” will appear on the terminal and nothing else will happen.

[This would be easy to fix, but then a document using the fix will silently fail on an older release of LATEX, missing all citations done with \nocite. Thus we do only generate an error message and leave the fix for a LATEX 2 $\varepsilon$  successor.]

Given that we are now a quarter century into using LATEX 2 $\varepsilon$  there is no good reason any more do limit ourself to 2.09 considerations. So we now simply delay the \nocite if it is issued in the preamble.

```
64 \ifx\@onlypreamble\document
```

Since we are after \begin{document} we can do the citations:

```
65 \@for\@citeb:=#1\do{%
66 \edef\@citeb{\expandafter\@firstofone\@citeb}%
67 \if@filesw\immediate\write\auxout{\string\citation{\@citeb}}\fi
68 \@ifundefined{b@\@citeb}{\G@refundefinedtrue
69 \@latex@warning{Citation ‘\@citeb’ undefined}}{}%
70 }
```

But before \begin{document} we raised an error message in the past but as of 2021/05 not any longer.

```
71 \% \@latex@error{Cannot be used in preamble}\@eha
```

Instead we delay the declaration to the start of the document. We have to use a late hook for this, so that it comes after the .aux file is open for writing and after \@preamblecmds was executed to change the above test. Therefore \AtBeginDocument would still be too early.

```
72 \AddToHook{begindocument/end}[kernel]{\nocite{#1}}%
73 \fi
```

```

74 \@esphack}
75 </2ekernel | latexrelease>
76 <latexrelease>\EndIncludeInRelease
77 <latexrelease>\IncludeInRelease{0000/00/00}%
78 <latexrelease> {\nocite}{Allow nocite in preamble}%
79 <latexrelease>
80 <latexrelease>\def\nocite#1{\@bsphack
81 <latexrelease> \ifx\@onlypreamble\document
82 <latexrelease> \@for\@citeb:=#1\do{%
83 <latexrelease> \edef\@citeb{\expandafter\@firstofone\@citeb}%
84 <latexrelease> \if@filesw\immediate\write\auxout{\string\citation{\@citeb}}\fi
85 <latexrelease> \@ifundefined{b@\@citeb}{\G@refundefinedtrue
86 <latexrelease> \G@refundefinedfalse
87 <latexrelease> \G@warning{Citation ‘\@citeb’ undefined}}{}%
88 <latexrelease> \else
89 <latexrelease> \G@latex@error{Cannot be used in preamble}\@eha
90 <latexrelease> \fi
91 <latexrelease> \@esphack}
92 <latexrelease>\EndIncludeInRelease
93 <*2ekernel>

```

Since `\nocite{*}` should not produce a warning about undefined citation keys (see PR 557), we need to set the control sequence ‘`\b@*`’ to something other than `\relax`. As a result `\cite{*}` will not warn either (but that never worked with BiBT<sub>E</sub>X in the first place).

```
94 \expandafter\let\csname b@\endcsname\empty
```

(*End definition for `\nocite`.*)

## 1.1 Default definitions

This hook determines the ‘relative formatting’ of the two logical parts of a citation with comment.

```
\@cite
95 \def\@cite#1#2{[#1\if@tempswa , #2\fi]}}

```

(*End definition for `\@cite`.*)

`\@cite@ofmt` This is, in general, a command that appears to have one argument whose value is, in the kernel, a single cs whose name is the expansion of `b@\@citeb`; the expansion of this cs will typically be some hmode material that produces the detailed typeset form of just the citations themselves.

```
96 \let\@cite@ofmt\hbox
```

(*End definition for `\@cite@ofmt`.*)

```
\@biblabel
97 \def\@biblabel#1{[#1]}
98 </2ekernel>
```

(*End definition for `\@biblabel`.*)

# File S

## ltpage.dtx

### 1 Page styles and related commands

#### 1.1 Page Style Commands

\pagestyle{\{style\}} : sets the page style of the current and succeeding pages to *style*  
\thispagestyle{\{style\}} : sets the page style of the current page only to *style*.  
To define a page style *style*, you must define \ps@*style* to set the page style parameters.

#### 1.2 How a page style makes running heads and feet

The \ps@... command defines the macros \oddhead, \oddfoot, \evenhead, and \evenfoot to define the running heads and feet. (See output routine.) To make headings determined by the sectioning commands, the page style defines the commands \chaptermark, \sectionmark, etc., where \chaptermark{\{text\}} is called by \chapter to set a mark. The \...mark commands and the \...head macros are defined with the help of the following macros.

(All the \...mark commands should be initialized to no-ops.)

#### 1.3 marking conventions

LATEX extends TEX's \mark facility by producing two kinds of marks a 'left' and a 'right' mark, using the following commands:

\markboth{\{left\}}{\{right\}} : Adds both marks.

\markright{\{right\}} : Adds a 'right' mark.

\leftmark : Used in the output routine, gets the current 'left' mark. Works like TEX's \botmark.

\rightmark : Used in the output routine, gets the current 'right' mark. Works like TEX's \firstmark. The marking commands work reasonably well for right marks 'numbered within' left marks—e.g., the left mark is changed by a \chapter command and the right mark is changed by a \section command. However, it does produce somewhat anomalous results if 2 \markboth's occur on the same page.

Commands like \tableofcontents that should set the marks in some page styles use a \cmkboth command, which is \let by the pagestyle command (\ps@...) to \markboth for setting the heading or to \gobbletwo to do nothing.

1 <\*2ekernel>

\pagestyle User command to set the page style for this and following pages.

```
2 \def\pagestyle#1{%
3 \@ifundefined{ps@#1}%
4 \undefinedpagestyle%
5 {\@nameuse{ps@#1}}}
```

(End definition for \pagestyle.)

\thispagestyle User command to set the page style for this page only.

```

6 \def\thispagestyle#1{%
7 \@ifundefined{ps@#1}%
8 \undefinedpagestyle
9 {\global\@specialpagetrue\gdef\@specialstyle{#1}}}

```

(End definition for \thispagestyle.)

\ps@empty The empty page style: No head or foot line.

```

10 \def\ps@empty{%
11 \let\@mkboth\@gobbletwo\let\@oddhead\@empty\let\@oddfoot\@empty
12 \let\@evenhead\@empty\let\@evenfoot\@empty}

```

(End definition for \ps@empty.)

\ps@plain The plain page style: No head, centred page number in foot.

```

13 \def\ps@plain{\let\@mkboth\@gobbletwo
14 \let\@oddhead\@empty\def\@oddfoot{\reset@font\hfil\thepage
15 \hfil}\let\@evenhead\@empty\let\@evenfoot\@oddfoot}

```

(End definition for \ps@plain.)

\@leftmark \@rightmark We implement \@leftmark and \@rightmark in terms of already defined commands to save token space. We can't get rid of them since they are sometimes used in applications.

```

16 \let\@leftmark\@firstoftwo
17 \let\@rightmark\@secondoftwo

```

(End definition for \@leftmark and \@rightmark.)

```

18 </2ekernel>
19 <*2ekernel | latexrelease>
20 <latexrelease>\IncludeInRelease{2019/10/01}%
21 <latexrelease> {\markboth}{Make commands robust}%

```

\markboth User commands for setting L<sup>A</sup>T<sub>E</sub>X marks.

\markright Test for \nobreak added 15 Apr 86 in \markboth and \markright letting \label and \index to \relax added 22 Feb 86 so these commands can appear in sectioning command arguments RmS 91/06/21 Same for \glossary

```

22 \DeclareRobustCommand*\markboth[2]{%
23 \begingroup
24 \let\label\relax \let\index\relax \let\glossary\relax
25 \unrestored@protected@xdef\@themark {{#1}{#2}}%
26 \temptokena \expandafter{\@themark}%
27 \mark{\the\temptokena}%
28 \endgroup
29 \ifnobreak\ifvmode\nobreak\fi\fi}

30 \DeclareRobustCommand*\markright[1]{%
31 \begingroup
32 \let\label\relax \let\index\relax \let\glossary\relax

```

Protection is handled inside \markright.

```

33 \expandafter\@markright\@themark {#1}%
34 \temptokena \expandafter{\@themark}%
35 \mark{\the\temptokena}%
36 \endgroup
37 \ifnobreak\ifvmode\nobreak\fi\fi}

```

```

(End definition for \markboth and \markright.)

38 </2ekernel | latexrelease>
39 <latexrelease>\EndIncludeInRelease
40 <latexrelease>\IncludeInRelease{0000/00/00}%
41 <latexrelease> {\markboth}{Make commands robust}%
42 <latexrelease>
43 <latexrelease>\kernel@make@fragile\markboth
44 <latexrelease>\kernel@make@fragile\markright
45 <latexrelease>
46 <latexrelease>\EndIncludeInRelease
47 {*2ekernel}

\@markright
\leftmark 48 \def\@markright#1#2{\@temptokena {#1}%
\rightmark 49 \unrestored@protected\xdef\@themark{{\the\@temptokena}{#3}}}
50 \def\leftmark{\expandafter\@leftmark\botmark\@empty\@empty}
51 \def\rightmark{\expandafter\@rightmark\firstmark\@empty\@empty}

(End definition for \@markright, \leftmark, and \rightmark.)

\@themark Initialise LATEX's marks without setting a TEX mark <whatsit>.
52 \def\@themark{}{ }

(End definition for \@themark.)

\mark Test versions of LATEX 2 ε initialised TEX's \mark system at this point, but this was
removed before the first release.

AtBeginDocument{\mark{}{}}

(End definition for \mark.)

\raggedbottom \raggedbottom typesets pages with no vertical stretch, so they have their natural height
instead of all being exactly the same height. (Uses a space of .0001fil to avoid interfering
with the 1fil space of \newpage.)
53 \DeclareRobustCommand\raggedbottom{%
54 \def\@textbottom{\vskip \z@ \oplus .0001fil}\let\@texttop\relax}

(End definition for \raggedbottom.)

\flushbottom \flushbottom: Inverse of \raggedbottom — makes all pages the same height.
55 \DeclareRobustCommand\flushbottom{%
56 \let\@textbottom\relax \let\@texttop\relax}

(End definition for \flushbottom.)

\sloppy \sloppy will never (well, hardly ever) produce overfull boxes, but may produce underfull
ones. (14 June 85)
57 \DeclareRobustCommand\sloppy{%
58 \tolerance 9999%
59 \emergencystretch 3em%
60 \hfuzz .5\p@
61 \vfuzz\hfuzz}

(End definition for \sloppy.)

```

**sloppypar** (*env.*) A `sloppypar` environment is equivalent to `{\par \sloppy ... \par}`.

```
62 \def\sloppypar{\par\sloppy}
63 \def\endsloppypar{\par}
```

**\fussy** Resets T<sub>E</sub>X's parameters to their normal finicky values.

```
64 \DeclareRobustCommand\fussy{%
65 \emergencystretch\z@
66 \tolerance 200%
67 \hfuzz .1\p@
68 \vfuzz\hfuzz}
```

(*End definition for \fussy.*)

**\overfullrule** L<sub>A</sub>T<sub>E</sub>X default is no overfull box rule. Changed by document class option.

```
69 \overfullrule Opt
```

(*End definition for \overfullrule.*)

```
70 </2ekernel>
```

# File T

## ltclass.dtx

### 1 Introduction

This file implements the following declarations, which replace `\documentstyle` in L<sup>A</sup>T<sub>E</sub>X 2<sub>E</sub> documents.

Note that old documents containing `\documentstyle` will be run using a compatibility option—thus keeping everyone happy, we hope!

The overall idea is that there are two types of ‘style files’: ‘class files’ which define elements and provide a default formatting for them; and ‘packages’ which provide extra functionality. One difference between L<sup>A</sup>T<sub>E</sub>X 2<sub>E</sub> and L<sup>A</sup>T<sub>E</sub>X 2.09 is that L<sup>A</sup>T<sub>E</sub>X 2<sub>E</sub> packages may have options. Note that options to classes packages may be implemented such that they input files, but these file names are not necessarily directly related to the option name.

### 2 User interface

```
\documentclass[⟨main-option-list⟩]{⟨class⟩}[⟨version⟩]
```

There must be exactly one such declaration, and it must come first. The *⟨main-option-list⟩* is a list of options which can modify the formatting of elements which are defined in the *⟨class⟩* file as well as in all following `\usepackage` declarations (see below). The *⟨version⟩* is a version number, beginning with a date in the format YYYY/MM/DD. If an older version of the class is found, a warning is issued.

```
\documentstyle[⟨main-option-list⟩]{⟨class⟩}[⟨version⟩]
```

The `\documentstyle` declaration is kept in order to maintain upward compatibility with L<sup>A</sup>T<sub>E</sub>X 2.09 documents. It is similar to `\documentclass`, but it causes all options in *⟨main-option-list⟩* that the *⟨class⟩* does not use to be passed to `\RequirePackage` after the options have been processed. This maintains compatibility with the 2.09 behaviour. Also a flag is set to indicate that the document is to be processed in L<sup>A</sup>T<sub>E</sub>X 2.09 compatibility mode. As far as most packages are concerned, this only affects the warnings and errors L<sup>A</sup>T<sub>E</sub>X generates. This flag does affect the definition of font commands, and `\sloppy`.

```
\usepackage[⟨package-option-list⟩]{⟨package-list⟩}[⟨version⟩]
```

There can be any number of these declarations. All packages in *⟨package-list⟩* are called with the same options.

Each *⟨package⟩* file defines new elements (or modifies those defined in the *⟨class⟩*), and thus extends the range of documents which can be processed. The *⟨package-option-list⟩* is a list of options which can modify the formatting of elements defined in the *⟨package⟩* file. The *⟨version⟩* is a version number, beginning with a date in the format YYYY/MM/DD. If an older version of the package is found, a warning is issued.

Each package is loaded only once. If the same package is requested more than once, nothing happens, unless the package has been requested with options that were not given the first time it was loaded, in which case an error is produced.

As well as processing the options given in the *⟨package-option-list⟩*, each package processes the *⟨main-option-list⟩*. This means that options that affect all of the packages can be given globally, rather than repeated for every package.

Note that class files have the extension `.cls`, packages have the extension `.sty`.

`filecontents` (*env.*)

The environment `filecontents` is intended for passing the contents of packages, options, or other files along with a document in a single file. It has one argument, which is the name of the file to create. If that file already exists (maybe only in the current directory if the OS supports a notion of a ‘current directory’ or ‘default directory’) then nothing happens (except for an information message) and the body of the environment is bypassed. Otherwise, the body of the environment is written verbatim to the file name given as the first argument, together with some comments about how it was produced.

The environment can also be called with an optional argument which is used to alter some of its behavior: option `force` or `overwrite` will allow for overwriting existing files, option `nosearch` will only check the current directory when looking if the file exists. This can be useful if you want to generate a local (modified) copy of some file that is already in the search tree of TeX. Finally, you can use `noheader` to prevent it from writing the standard blurb at the top of the file (this is actually the same as using the star form of the environment).

The environment is now allowed anywhere in the document, but to ensure that all packages or options necessary are available when the document is run, it is normally best to place it at the top of your file (before `\documentclass`). A possible use case for using it inside the document body is if you want to reuse some text several times in the document you could then write it and later use `\input` to retrieve it where needed.

The begin and end tags should each be on a line by itself.

## 2.1 Option processing

When the options are processed, they are divided into two types: *local* and *global*:

- For a class, the options in the `\documentclass` command are local.
- For a package, the options in the `\usepackage` command are local, and the options in the `\documentclass` command are global.

The options for `\documentclass` and `\usepackage` are processed in the following way:

1. The local and global options that have been declared (using `\DeclareOption` as described below) are processed first.

In the case of `\ProcessOptions`, they are processed in the order that they were declared in the class or package.

In the case of `\ProcessOptions*`, they are processed in the order that they appear in the option-lists. First the global options, and then the local ones.

2. Any remaining local options are dealt with using the default option (declared using the `\DeclareOption*` declaration described below). For document classes, this usually does nothing, but records the option on a list of unused options. For packages, this usually produces an error.

Finally, when `\begin{document}` is reached, if there are any global options which have not been used by either the class or any package, the system will produce a warning.

## 3 Class and Package interface

### 3.1 Class name and version

`\ProvidesClass` A class can identify itself with the `\ProvidesClass{\name}{\version}` command. The `\version` should begin with a date in the format YYYY/MM/DD.

### 3.2 Package name and version

`\ProvidesPackage` A package can identify itself with the `\ProvidesPackage{\name}{\version}` command. The `\version` should begin with a date in the format YYYY/MM/DD.

### 3.3 Requiring other packages

`\RequirePackage` Packages or classes can load other packages using `\RequirePackage[\options]{\name}{\version}`.

If the package has already been loaded, then nothing happens unless the requested options are not a subset of the options with which it was loaded, in which case an error is called.

`\LoadClass` Similar to `\RequirePackage`, but for classes, may not be used in package files.

`\PassOptionsToPackage` Packages can pass options to other packages using:

`\PassOptionsToPackage{\options}{\package}`.

`\PassOptionsToClass` This adds the `\options` to the options list of any future `\RequirePackage` or `\usepackage` command. For example:

```
\PassOptionsToPackage{foo,bar}{fred}
```

is the same as:

```
\RequirePackage[foo,bar,baz]{fred}
```

`\LoadClassWithOptions` `\LoadClassWithOptions{\name}{\version}`:

This is similar to `\LoadClass`, but it always calls class `\name` with exactly the same option list that is being used by the current class, rather than an option explicitly supplied or passed on by `\PassOptionsToClass`. `\RequirePackageWithOptions` is the analogous command for packages.

This is mainly intended to allow one class to simply build on another, for example:

```
\LoadClassWithOptions{article}
```

This should be contrasted with the slightly different construction

```
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions
\LoadClass{article}
```

As used here, the effects are more or less the same, but the version using `\LoadClassWithOptions` is slightly quicker (and less to type). If, however, the class declares options of its own then the two constructions are different; compare, for example:

```
\DeclareOption{landscape}{...}
\ProcessOptions
\LoadClassWithOptions{article}
```

with:

```
\DeclareOption{landscape}{...}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions
\LoadClass{article}
```

In the first case, the `article` class will be called with option `landscape` precisely when the current class is called with this option; but in the second example it will not as in that case `article` is only passed options by the default option handler, which is not used for `landscape` as that option is explicitly declared.

`\IfPackageLoadedTF` To find out if a package has already been loaded, use  
`\IfClassLoadedTF` `\IfPackageLoadedTF{(package)}{<true>}{<false>}`  
`\@ifpackageloaded`  
`\@ifclassloaded`  
or the old name `\@ifpackageloaded`.  
`\IfPackageAtLeastTF` To find out if a package has already been loaded with a version equal to or more recent than `<date>`, use  
`\IfClassAtLeastTF` `\IfPackageAtLeastTF{(package)}{<date>}{{<true>}{<false>}}`  
`\@ifpackagelater`  
`\@ifclasslater`  
or the old name `\@ifpackagelater`.  
`\IfFormatAtLeastTF` To test the format date use  
`\IfFormatAtLeastTF{(date)}{<true>}{<false>}`

`\IfPackageLoadedWithOptionsTF` To find out if a package has already been loaded with at least the options `<options>`,  
`\IfClassLoadedWithOptionsTF` use  
`\@ifpackagewith` `\IfPackageLoadedWithOptionsTF{(package)}{<options>}{{<true>}{<false>}}`  
`\@ifclasswith`

or the old name `\@ifpackagewith`.

There exists one package that can't be tested with the above commands: the `fontenc` package pretends that it was never loaded to allow for repeated reloading with different options (see `ltoutenc.dtx` for details).

### 3.4 Declaring new options

Options for classes and packages are built using the same macros.

`\DeclareOption` To define a builtin option, use `\DeclareOption{<name>}{<code>}`.  
`\DeclareOption*` To define the default action to perform for local options which have not been declared, use `\DeclareOption*{<code>}`.  
*Note:* there should be no use of  
`\RequirePackage`, `\DeclareOption`, `\DeclareOption*` or `\ProcessOptions` inside `\DeclareOption` or `\DeclareOption*`.  
Possible uses for `\DeclareOption*` include:  
`\DeclareOption*{}`  
Do nothing. Silently accept unknown options. (This suppresses the usual warnings.)  
`\DeclareOption*{\@unkownoptionerror}`  
Complain about unknown local options. (The initial setting for package files.)  
`\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{<pkg-name>}}`  
Handle the current option by passing it on to the package `<pkg-name>`, which will presumably be loaded via `\RequirePackage` later in the file. This is useful for building

‘extension’ packages, that perhaps handle a couple of new options, but then pass everything else on to an existing package.

```
\DeclareOption*{\InputIfFileExists{xx-\CurrentOption.yyy}%
{}%
{\OptionNotUsed}}
```

Handle the option `foo` by loading the file `xx-foo.yyy` if it exists, otherwise do nothing, but declare that the option was not used. Actually the `\OptionNotUsed` declaration is only needed if this is being used in class files, but does no harm in package files.

### 3.5 Safe Input Macros

|                                 |                                                                                                                                                                                                                                                                                                                                                               |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\InputIfFileExists</code> | <code>\InputIfFileExists{\langle file \rangle}{\langle then \rangle}{\langle else \rangle}</code>                                                                                                                                                                                                                                                             |
|                                 | Inputs <code>\langle file \rangle</code> if it exists. Immediately before the input, <code>\langle then \rangle</code> is executed. Otherwise <code>\langle else \rangle</code> is executed.                                                                                                                                                                  |
| <code>\IfExists</code>          | As above, but does not input the file.                                                                                                                                                                                                                                                                                                                        |
|                                 | One thing you might like to put in the <code>\langle else \rangle</code> clause is                                                                                                                                                                                                                                                                            |
| <code>\@missingfileerror</code> | This starts an interactive request for a filename, supplying default extensions. Just hitting return causes the whole input to be skipped and entering <code>x</code> quits the current run,                                                                                                                                                                  |
| <code>\input</code>             | This has been redefined from the L <sup>A</sup> T <sub>E</sub> X2.09 definition, in terms of the new commands <code>\InputIfFileExists</code> and <code>\@missingfileerror</code> .                                                                                                                                                                           |
| <code>\listfiles</code>         | Giving this declaration in the preamble causes a list of all files input via the ‘safe input’ commands to be listed at the end. Any strings specified in the optional argument to <code>\ProvidesPackage</code> are listed alongside the file name. So files in standard (and other non-standard) distributions can put informative strings in this argument. |

## 4 Implementation

|                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                  | <code>1 \begin{*2ekernel}</code>                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>\if@compatibility</code>   | The flag for compatibility mode.                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|                                  | <code>2 \newif\if@compatibility</code>                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|                                  | <i>(End definition for \if@compatibility.)</i>                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <code>\@documentclasshook</code> | This legacy hook is called after the first <code>\documentclass</code> command. It is <i>not</i> integrated with the new 2020 hook management system! By default this checks to see if <code>\@normalsize</code> is undefined, and if so, sets it to <code>\normalsize</code> .                                                                                                                                                                                                |
|                                  | <code>3 \def\@documentclasshook{% 4   \ifx\@normalsize\@undefined 5     \let\@normalsize\normalsize 6   \fi 7 }</code>                                                                                                                                                                                                                                                                                                                                                         |
|                                  | <i>(End definition for \@documentclasshook.)</i>                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>\@declaredoptions</code>   | This list is automatically built by <code>\DeclareOption</code> . It is the list of options (separated by commas) declared in the class or package file and it defines the order in which the corresponding <code>\ds@\langle option \rangle</code> commands are executed. All local <code>\langle option \rangle</code> s which are not declared will be processed in the order defined by the optional argument of <code>\documentclass</code> or <code>\usepackage</code> . |
|                                  | <code>8 \let\@declaredoptions\empty</code>                                                                                                                                                                                                                                                                                                                                                                                                                                     |

(End definition for \@declaredoptions.)

\@classoptionslist List of options of the main class.  
 9 \let\@classoptionslist\relax  
 10 %\onlypreamble\@classoptionslist  
 (End definition for \@classoptionslist.)

\@raw@classoptionslist List of options of the main class (unprocessed).  
 11 \let\@raw@classoptionslist\relax  
 (End definition for \@raw@classoptionslist.)

\@unusedoptionlist List of options of the main class that haven't been declared or loaded as class option files.  
 12 \let\@unusedoptionlist\@empty  
 13 %\onlypreamble\@unusedoptionlist  
 (End definition for \@unusedoptionlist.)

\CurrentOption Name of current package or option.  
 14 \let\CurrentOption\@empty  
 (End definition for \CurrentOption.)

\@currpath Path to the current file if explicitly given.  
 15 </2ekernel>  
 16 <\*2ekernel | latexrelease>  
 17 <latexrelease>  
 18 <latexrelease>\IncludeInRelease{2020/10/01}{\@currpath}%  
 19 <latexrelease> {Add \@currpath}%  
 20 \let\@currpath\@empty  
 21 <latexrelease>\EndIncludeInRelease  
 22 %  
 23 <latexrelease>\IncludeInRelease{0000/00/00}{\@currpath}%  
 24 <latexrelease> {Add \@currpath}%  
 25 <latexrelease>\let\@currpath\@undefined  
 26 <latexrelease>\EndIncludeInRelease  
 27 </2ekernel | latexrelease>  
 28 <\*2ekernel>  
 (End definition for \@currpath.)

\@currname Name of current package or option.  
 29 \let\@currname\@empty  
 (End definition for \@currname.)

\@currext The current file extension.  
 30 \global\let\@currext=\@empty  
 (End definition for \@currext.)

\@clsextension The two possible values of \@currext.  
 \@pkgextension  
 31 \def\@clsextension{cls}  
 32 \def\@pkgextension{sty}

(End definition for \c@lsextension and \c@pkgeextension.)

```
\c@pushfilename Commands to push and pop the file name and extension.
\c@popfilename #1 current name.
\c@currnamestack #2 current extension.
#3 current catcode of @.
#4 Rest of the stack.
33 </2ekernel>
34 {*2ekernel | latexrelease}
35 {latexrelease}
36 {latexrelease}\IncludeInRelease{2020/10/01}{\c@pushfilename}%
37 {latexrelease} {Add \c@expl@push@filename@@ and \c@expl@push@filename@aux@@}%
38 {\def\c@pushfilename{%
```

The push and pop macros are injected in \c@pushfilename and \c@popfilename so that they correctly keep track of the hook labels.

This needs cleanup with the expl3 interfaces also playing here, e.g., \c@expl@push@filename@@ needs cleanup and (and should probably not have this name either).

```
39 \c@expl@push@filename@@
40 \xdef\c@currnamestack{
41 {\\@currname}%
42 {\\@currext}%
43 {\\the\\catcode`\\@}%
44 \\@currnamestack}%
```

Temporarily add a stack for \c@currpath here. This should be integrated in the main file stack eventually, but other packages rely on \c@currnamestack having three elements per file, so that isn't a trivial change. The prefix \c@kernel@... hopefully discourages people from using it.

```
45 \xdef\\@kernel@currpathstack{
46 {\\@currpath}%
47 \\@kernel@currpathstack}%
48 \\@expl@push@filename@aux@@}
49 {\\@kernel@currpathstack}%
```

The following version of \c@pushfilename didn't formally exist in this file, but in the 2020/02/02 release, expl3 was preloaded and it patched \c@pushfilename (and \c@popfilename) by adding some hooks in there. But rolling back to 2020/02/02, expl3 doesn't patch these macros again, so rolling back has to take those hooks into account. Same goes for \c@popfilename.

```
50 {\\@kernel@currpathstack}%
51 {\\@kernel@currpathstack}%
52 {\\@kernel@currpathstack}%
53 {\\@kernel@currpathstack}%
54 {\\@kernel@currpathstack}%
55 {\\@kernel@currpathstack}%
56 {\\@kernel@currpathstack}%
57 {\\@kernel@currpathstack}%
58 {\\@kernel@currpathstack}%
59 {\\@kernel@currpathstack}%
60 {\\@kernel@currpathstack}%
61 {\\@kernel@currpathstack}%
62 {\\@kernel@currpathstack}%
```

When we roll back from a release that has `\expl3` preloaded, the definitions of `\@pushfilename` and `\@popfilename` can't be completely rolled back otherwise `\ExplSyntaxOff` based packages won't have the automatic `\ExplSyntaxOff` at the end. Here and below for `\@popfilename`, we don't roll back all the way through if coming from L<sup>A</sup>T<sub>E</sub>X > 2020 – 02 – 02.

```

63 ⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@pushfilename}%
64 ⟨latexrelease⟩ {Add \Expl@push@filename@@ and \Expl@push@filename@aux@@}%
65 ⟨latexrelease⟩\ifnum\sourceLaTeXdate<20200202\relax
66 ⟨latexrelease⟩ \GenericInfo{}{Defining 00-00-00\string\@pushfilename.}
67 ⟨latexrelease⟩\def\@pushfilename{%
68 ⟨latexrelease⟩ \xdef\@currnamestack{%
69 ⟨latexrelease⟩ {\@currname}%
70 ⟨latexrelease⟩ {\@currext}%
71 ⟨latexrelease⟩ {\the\catcode`\@}%
72 ⟨latexrelease⟩ {\@currnamestack}%
73 ⟨latexrelease⟩\else
74 ⟨latexrelease⟩ \GenericInfo{}{Defining 2020-02-02\string\@pushfilename.}
75 ⟨latexrelease⟩\def\@pushfilename{%
76 ⟨latexrelease⟩ \Expl@push@filename@@
77 ⟨latexrelease⟩ \xdef\@currnamestack{%
78 ⟨latexrelease⟩ {\@currname}%
79 ⟨latexrelease⟩ {\@currext}%
80 ⟨latexrelease⟩ {\the\catcode`\@}%
81 ⟨latexrelease⟩ {\@currnamestack}%
82 ⟨latexrelease⟩ {\Expl@push@filename@aux@@}
83 ⟨latexrelease⟩\fi
84 ⟨latexrelease⟩\EndIncludeInRelease
85 \@onlypreamble\@pushfilename

86 ⟨latexrelease⟩
87 ⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\@popfilename}%
88 ⟨latexrelease⟩ {Add \Expl@pop@filename@@}%
89 \def\@popfilename{\Expl@@hook@curr@name@pop@@
90 \expandafter\@p@filename\@currnamestack\@nil

```

Same for popping:

```

91 \expandafter\@p@filepath\@kernel@currpathstack\@nil
92 \Expl@pop@filename@@
93 ⟨latexrelease⟩\EndIncludeInRelease
94 ⟨latexrelease⟩
95 ⟨latexrelease⟩\IncludeInRelease{2020/02/02}{\@popfilename}%
96 ⟨latexrelease⟩ {Add \Expl@push@filename@@}%
97 ⟨latexrelease⟩\def\@popfilename{\expandafter\@p@filename\@currnamestack\@nil
98 ⟨latexrelease⟩ {\Expl@pop@filename@@}
99 ⟨latexrelease⟩\EndIncludeInRelease
100 ⟨latexrelease⟩

101 ⟨latexrelease⟩\IncludeInRelease{0000/00/00}{\@popfilename}%
102 ⟨latexrelease⟩ {Add \Expl@push@filename@@ and \Expl@push@filename@aux@@}%
103 ⟨latexrelease⟩\ifnum\sourceLaTeXdate<20200202\relax
104 ⟨latexrelease⟩ \GenericInfo{}{Defining 00-00-00\string\@popfilename.}
105 ⟨latexrelease⟩\def\@popfilename{\expandafter\@p@filename\@currnamestack\@nil}
106 ⟨latexrelease⟩\else
107 ⟨latexrelease⟩ \GenericInfo{}{Defining 2020-02-02\string\@popfilename.}

```

```

108 \def\@popfilename{\expandafter\@p@filename\@currnamestack\@nil
109 \def\@popfilename{\expandafter\@empty\@currnamestack\@nil}
110 \fi
111 \EndIncludeInRelease
112 \onlypreamble\@popfilename
113 {/2ekernel | latexrelease}
114 {*2ekernel}
115 \def\@p@filename#1#2#3#4\@nil{%
116 \gdef\@currname{#1}%
117 \gdef\@currext{#2}%
118 \catcode`\@#3\relax
119 \gdef\@currnamestack{#4}%
120 \onlypreamble\@p@filename
121 \gdef\@currnamestack{}%
122 \onlypreamble\@currnamestack

```

(End definition for `\@pushfilename`, `\@popfilename`, and `\@currnamestack`.)

`\@kernel@\currpathstack` Path to the current file if explicitly given. The auxiliary is needed here to insert a `\@empty` to prevent the loss of braces.

```

123 {/2ekernel}
124 {*2ekernel | latexrelease}
125 \def\@currpathstack{\@kernel@\currpathstack}%
126 \IncludeInRelease{2020/10/01}{\@kernel@\currpathstack}%
127 {\Add\@kernel@\currpathstack}%

```

If rolling backwards to this release, `\@kernel@\currpathstack` will be defined, so the `\gdef` line should not be executed, thus the `\@gobblethree` will take it out, so the stack isn't touched.

```

128 \IfUndefined{\@kernel@\currpathstack}{}{\@gobblethree}
129 \gdef\@kernel@\currpathstack{}%

```

If rolling forward to this release, then the `\gdef` line above will define the path stack to be empty (which it can't be, inside a file), so the code below will traverse the `\@currnamestack`, and add as many empty items to `\@kernel@\currpathstack` as there are items in `\@currnamestack`, so both are back in sync. Most of the time `latexrelease` is loaded on top-level, so only one item is needed, but `platexrelease` loads it internally, so the more complicated loop is needed.

```

130 \ifx\@kernel@\currpathstack\@empty
131 \def\reserved@a#1#2#3{%
132 \ifx\relax#3\else
133 \g@addto@macro\@kernel@\currpathstack{\{}%
134 \expandafter\reserved@a
135 \fi}%
136 \expandafter\reserved@a\@currnamestack{\{}{\relax}%
137 \fi
138 \def\@p@filepath#1{%
139 \gdef\@currpath{\#1}\@p@filepath@aux\@empty}
140 \def\@p@filepath@aux#1\@nil{%
141 \xdef\@kernel@\currpathstack{\#1}%
142 \EndIncludeInRelease
143 %
144 \IncludeInRelease{0000/00/00}{\@kernel@\currpathstack}%

```

```

145 \let\@kernel@currpathstack\@undefined
146 \let\@kernel@currpathstack\@undefined
147 \let\@p@filepath\@undefined
148 \let\@p@filepath@aux\@undefined
149 \EndIncludeInRelease
150 {/2ekernel | latexrelease}
151 {*2ekernel}

```

(End definition for `\@kernel@currpathstack`.)

`\@optionlist` Returns the option list of the file.

```

152 \def\@optionlist#1{%
153 \@ifundefined{opt@#1}\@empty{\csname opt@#1\endcsname}%
154 \%@\onlypreamble\@optionlist

```

(End definition for `\@optionlist`.)

`\@ifpackageloaded` `\@ifpackageloaded{<name>}` Checks to see whether a file has been loaded.

```

\@ifclassloaded 155 \def\@ifpackageloaded{\@ifl@aded\@pkgextension}
156 \def\@ifclassloaded{\@ifl@aded\@clsextension}
157 \def\@ifl@aded#1#2{%
158 \expandafter\ifx\csname ver@#2.#1\endcsname\relax
159 \expandafter\@secondoftwo
160 \else
161 \expandafter\@firstoftwo
162 \fi}

```

(End definition for `\@ifpackageloaded` and `\@ifclassloaded`.)

`\@ifpackagelater` `\@ifpackagelater{<name>}{YYYY/MM/DD}{<true code>}{<false code>}` Checks that the package loaded is more recent or equal to the given date. A better name for it would therefore been `\@ifpackagelaterorequal` but it is in use for more than 30 years, so ...

```

163 \def\@ifpackagelater{\@ifl@ter\@pkgextension}
164 \def\@ifclasslater{\@ifl@ter\@clsextension}

```

(End definition for `\@ifpackagelater` and `\@ifclasslater`.)

`\IfPackageAtLeastTF` `\IfFormatAtLeastTF{YYYY/MM/DD}{<true code>}{<false code>}` Test if the format is later or equal to the given date.

```

\IfClassAtLeastTF 165 {/2ekernel}
166 {*2ekernel | latexrelease}
167 \let\@kernel@currpathstack\@undefined
168 \let\@p@filepath\@undefined
169 \let\@p@filepath@aux\@undefined
170 \let\@kernel@currpathstack\@undefined
171 \let\@p@filepath\@undefined
172 \let\@p@filepath@aux\@undefined

```

For rollback pretend it was available since the beginning of dawn.

```

173 \let\@kernel@currpathstack\@undefined
174 \let\@p@filepath\@undefined
175 \let\@p@filepath@aux\@undefined
176 \let\@kernel@currpathstack\@undefined
177 \let\@p@filepath\@undefined
178 \let\@p@filepath@aux\@undefined

```

```

179 ⟨latexrelease⟩\EndIncludeInRelease
180 ⟨*2ekernel⟩

(End definition for \IfPackageAtLeastTF, \IfClassAtLeastTF, and \IfFormatAtLeastTF.)
```

\@ifl@ter

```

181 \def\@ifl@ter#1#2{%
182 \expandafter\@ifl@t@r
183 \csname ver@#2.#1\endcsname}
184 ⟨/2ekernel⟩

This internal macro is also used in \NeedsTeXFormat.

185 ⟨latexrelease⟩\IncludeInRelease{2018/04/01}%
186 ⟨latexrelease⟩ {\@ifl@t@r}{Guard against bad input}%
187 ⟨*2ekernel | latexrelease⟩
188 \def\@ifl@t@r#1#2{%
189 \ifnum\expandafter\@parse@version@#1//00@nil<%
190 \expandafter\@parse@version@#2//00@nil
191 \expandafter\@secondoftwo
192 \else
193 \expandafter\@firstoftwo
194 \fi}
195 \def\@parse@version@#1{\@parse@version0#1}
196 ⟨/2ekernel | latexrelease⟩
197 ⟨latexrelease⟩\EndIncludeInRelease
198 ⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
199 ⟨latexrelease⟩ {\@ifl@t@r}{Guard against bad input}%
200 ⟨latexrelease⟩\def\@ifl@t@r#1#2{%
201 \ifnum\expandafter\@parse@version#1//00@nil<%
202 \expandafter\@parse@version#2//00@nil
203 \expandafter\@secondoftwo
204 \else
205 \expandafter\@firstoftwo
206 \fi}
207 ⟨latexrelease⟩\let\@parse@version@\undefined
208 ⟨latexrelease⟩\EndIncludeInRelease
209 ⟨*2ekernel⟩

(End definition for \@ifl@ter.)
```

210 ⟨/2ekernel⟩
211 ⟨\*2ekernel | latexreleasefirst⟩
212 \def\@parse@version#1/#2/#3#4#5@nil{%
213 \@parse@version@dash#1-#2-#3#4@nil
214 }

The \if test here ensures that an argument with no / or - produces 0 (actually 00).

```

215 \def\@parse@version@dash#1-#2-#3#4#5@nil{%
216 \if\relax#2\relax\else#1\fi#2#3#4 }
217 ⟨/2ekernel | latexreleasefirst⟩
218 ⟨*2ekernel⟩
```

\@ifpackagewith \@ifpackagewith{⟨name⟩}{⟨option-list⟩} Checks that ⟨option-list⟩ is a subset of the options **with** which ⟨name⟩ was loaded.

```

219 \def\@ifpackagewith{\@ifoptions\@pkgextension}
220 \def\@ifclasswith{\@ifoptions\@clsextension}
```

```

221 \def\@if@options#1#2{%
222 \@expandtwoargs\@if@pti@ns{\@optionlist{#2.#1}}}
223 Probably shouldn't use \CurrentOption here... (changed to \reserved@b.)
224 </2ekernel>
225 <latexrelease>\IncludeInRelease{2017/01/01}%
226 {\@if@pti@ns}{Spaces in option clash check}%
227 <*2ekernel | latexrelease>
228 \def\@if@pti@ns#1#2{%
229 \let\reserved@a\@firstoftwo
230 \edef\reserved@b{\zap@space#2 \empty}%
231 \for\reserved@b:=\reserved@b\do{%
232 \ifx\reserved@b\empty
233 \else
234 \expandafter\in@\expandafter{\expandafter,\reserved@b,}{, #1,}%
235 \ifin@
236 \else
237 \let\reserved@a\@secondoftwo
238 \fi
239 }%
240 \reserved@a
241 </2ekernel | latexrelease>
242 <latexrelease>\EndIncludeInRelease
243 <latexrelease>\IncludeInRelease{0000/00/00}%
244 {\@if@pti@ns}{Spaces in option clash check}%
245 <latexrelease>\def\@if@pti@ns#1#2{%
246 \let\reserved@a\@firstoftwo
247 \for\reserved@b:=#2\do{%
248 \ifx\reserved@b\empty
249 \else
250 \expandafter\in@\expandafter
251 {\expandafter,\reserved@b,}{, #1,}%
252 \ifin@
253 \else
254 \let\reserved@a\@secondoftwo
255 \fi
256 }%
257 }%
258 \reserved@a
259 <latexrelease>\EndIncludeInRelease
260 <*2ekernel>

```

(End definition for \@ifpackagewith and \@ifclasswith.)

\IfPackageLoadedTF More public names for the commands already available for a long time.

```

\IfPackageLoadedWithOptionsTF
\IfClassLoadedTF
\IfClassLoadedWithOptionsTF
261 </2ekernel>
262 <*2ekernel | latexrelease>
263 <latexrelease>\IncludeInRelease{2021/11/15}%
264 {\IfPackageLoadedTF}{Test package loading}%
265 \let \IfPackageLoadedTF \@ifpackageloaded
266 \let \IfClassLoadedTF \@ifclassloaded
267 \let \IfPackageLoadedWithOptionsTF \@ifpackagewith
268 \let \IfClassLoadedWithOptionsTF \@ifclasswith

```

For rollback pretend it was available since the beginning of dawn.

```
269 </2ekernel | latexrelease>
270 <latexrelease>\EndIncludeInRelease
271 <latexrelease>\IncludeInRelease{0000/00/00}%
272 <latexrelease> {\IfPackageLoadedtTF}{Test package loading}%
273 <latexrelease>
274 <latexrelease>\let \IfPackageLoadedTF \@ifpackageloaded
275 <latexrelease>\let \IfClassLoadedTF \@ifclassloaded
276 <latexrelease>\let \IfPackageLoadedWithOptionsTF \@ifpackagewith
277 <latexrelease>\let \IfClassLoadedWithOptionsTF \@ifclasswith
278 <latexrelease>
279 <latexrelease>\EndIncludeInRelease
280 <*2ekernel>

(End definition for \IfPackageLoadedTF and others.)
```

\ProvidesPackage Checks that the current filename is correct, and defines \ver@filename.

```
281 </2ekernel>
282 <latexrelease>\IncludeInRelease{2020/10/01}%
283 <latexrelease> {\ProvidesPackage}{Check name with \strcmp}%
284 <*2ekernel | latexrelease>
285 \def\ProvidesPackage#1{%
286 \xdef\@gtempa{#1}%

```

Here \@currpath is explicitly added to the file name to report when a package or class is loaded using an explicit path. Loading using a path in the argument is supported but not encouraged.

```
287 \@expandtwoargs\@expl@str@if@eq@nnTF
288 {\@gtempa}{\@currpath\@currname}{}{%
289 \@latex@warning@no@line{You have requested
290 \@cls@pkg\space`\@currpath\@currname',\MessageBreak
291 but the \@cls@pkg\space provides '#1'}%
292 }%
293 \@ifnextchar[\@pr@videopackage{\@pr@videopackage[]}]%
294 \onlypreamble\ProvidesPackage
295 </2ekernel | latexrelease>
296 <latexrelease>\EndIncludeInRelease
297 %
298 <latexrelease>\IncludeInRelease{0000/00/00}%
299 <latexrelease> {\ProvidesPackage}{Undo: check name with \strcmp}%
300 <latexrelease>\def\ProvidesPackage#1{%
301 <latexrelease> \xdef\@gtempa{#1}%
302 <latexrelease> \ifx\@gtempa\@currname\else
303 <latexrelease> \@latex@warning@no@line{You have requested
304 <latexrelease> \@cls@pkg\space`\@currname',\MessageBreak
305 <latexrelease> but the \@cls@pkg\space provides '#1'}%
306 <latexrelease> \fi
307 <latexrelease> \@ifnextchar[\@pr@videopackage{\@pr@videopackage[]}]%
308 <latexrelease>\EndIncludeInRelease
309 <*2ekernel>
```

(End definition for \ProvidesPackage.)

\@pr@videopackage This is the helper command for \ProvidesPackage. It tries to be cautious when handling the identification string in case it contains UTF-8 characters.

```

310 </2ekernel>
311 <*2ekernel | latexrelease>
312 <latexrelease>\IncludeInRelease{2020/10/01}%
313 <latexrelease> {\@pr@videopackage}{Allow for package substitution}%
314 \def\@pr@videopackage[#1]{%
315 \expandafter\protected@xdef % <-- protected...
316 \csname ver@\@currname.\@currext\endcsname{#1}\% Loaded package
317 \expandafter\let
318 \csname ver@\@currpkg@reqd\expandafter\endcsname % Requested package
319 \csname ver@\@currname.\@currext\endcsname
320 \ifx\@currext\@clsextension
321 \typeout{Document Class: \@gtempa\space#1}%
322 \else
323 \protected@wlog{Package: \@gtempa\space#1}\% <--- protected
324 \fi}

(End definition for \@pr@videopackage.)
```

\protected@wlog This is like plain T<sub>E</sub>X's \wlog but gracefully handles protected commands.

```

325 \long\def\protected@wlog#1{\begingroup
326 \set@display@protect
327 \immediate \write \m@ne {#1}\endgroup }

(End definition for \protected@wlog.)

328 </2ekernel | latexrelease>
329 <latexrelease>\EndIncludeInRelease
330 <latexrelease>\IncludeInRelease{2020/02/02}%
331 <latexrelease> {\@pr@videopackage}{Protection for package info}%
332 <latexrelease>
333 <latexrelease>\def\@pr@videopackage[#1]{%
334 \expandafter\protected@xdef % <-- protected...
335 \csname ver@\@currname.\@currext\endcsname{#1}\%
336 \ifx\@currext\@clsextension
337 \typeout{Document Class: \@gtempa\space#1}%
338 \else
339 \protected@wlog{Package: \@gtempa\space#1}\% <--- protected
340 \fi}
341 <latexrelease>
342 <latexrelease>\EndIncludeInRelease
343 <latexrelease>\IncludeInRelease{0000/00/00}%
344 <latexrelease> {\@pr@videopackage}{Protection for package info}%
345 <latexrelease>
346 <latexrelease>\def\@pr@videopackage[#1]{%
347 \expandafter\xdef\csname ver@\@currname.\@currext\endcsname{#1}\%
348 \ifx\@currext\@clsextension
349 \typeout{Document Class: \@gtempa\space#1}%
350 \else
351 \wlog{Package: \@gtempa\space#1}\%
352 \fi}
353 <latexrelease>\let\protected@wlog\@undefined
354 <latexrelease>\EndIncludeInRelease
355 <*2ekernel>
```

```
357 \onlypreamble\prvidepackage
```

- \ProvidesClass Like \ProvidesPackage, but for classes. This needs a dummy `latexrelease` block to copy the definition of \ProvidesPackage as it changes across releases.

```
358 {/2ekernel}
359 <latxrelease>\IncludeInRelease{0000/00/00}%
360 <latxrelease> {\ProvidesClass}{Track \ProvidesPackage}%
361 {*2ekernel | latxrelease}
362 \let\ProvidesClass\ProvidesPackage
363 \onlypreamble\ProvidesClass
364 {/2ekernel | latxrelease}
365 <latxrelease>\EndIncludeInRelease
366 {*2ekernel}
```

(End definition for \ProvidesClass.)

- \ProvidesFile Like \ProvidesPackage, but for arbitrary files. Do not apply \onlypreamble to these, as we may want to label files input during the document.

```
\@providesfile 367 \def\ProvidesFile#1{%
368 \begingroup
369 \catcode`\ 10 %
370 \ifnum \endlinechar<256 %
371 \ifnum \endlinechar>\m@ne
372 \catcode\endlinechar 10 %
373 \fi
374 \fi
375 \makeother\%
376 \makeother\&%
377 }
```

```
\kernel@ifnextchar[{\@providesfile{#1}}{\@providesfile{#1}[]}]}
```

During initex a special version of \@providesfile is used. The real definition is installed right at the end, in `ltfinal.dtx`.

```
def\@providesfile#1[#2]{%
 \wlog{File: #1 #2}%
 \expandafter\xdef\csname ver@#1\endcsname{#2}%
}
```

(End definition for \ProvidesFile and \@providesfile.)

- \PassOptionsToPackage If the package has been loaded, we check that it was first loaded with the options.  
\PassOptionsToClass Otherwise we add the option list to that of the package.

```
378 {/2ekernel}
379 <latxrelease>\IncludeInRelease{2021/06/01}%
380 <latxrelease> {\@passoptions}{Raw option lists}%
381 {*2ekernel | latxrelease}
382 \def\@passoptions#1#2#3{%
383 \expl@@@filehook@set@curr@file@nN
384 {\expl@@@filehook@resolve@file@subst@w #3.#1@nil}%
385 \reserved@a\reserved@b
386 \expl@@@filehook@clear@replacement@flag@%
387 \expandafter\xdef\csname opt@\reserved@a\endcsname{%
```

```

388 \@ifundefined{opt@\reserved@a}\empty%
389 {\csname opt@\reserved@a\endcsname,}%
390 \zap@space#2 \empty%
391
392 \expandafter\let
393 \csname opt@#3.#1\expandafter\endcsname
394 \csname opt@\reserved@a\endcsname
395
396 Extend raw option list
397
398 \ifundefined{@raw@opt@#3.#1}%
399 {\expandafter\gdef\csname @raw@opt@#3.#1\expandafter\endcsname
400 \expandafter{\#2}}%
401 {\expandafter\gaddto@macro\csname @raw@opt@#3.#1\expandafter\endcsname
402 \expandafter{\expandafter,\#2}}%
403 }
404
405 \if2ekernel | latexrelease>
406 \EndIncludeInRelease
407
408 \iflatexrelease\IncludeInRelease{2020/10/01}{\@pass@ptions}
409 \iflatexrelease {Add file replacement in \@pass@ptions}%
410 \iflatexrelease
411 \iflatexrelease\def\@pass@ptions#1#2#3{%
412 \expl@@@filehook@set@curr@file@@nN
413 {\expl@@@filehook@resolve@file@subst@w #3.#1\@nil}%
414 \reserved@a\reserved@b
415 \expl@@@filehook@clear@replacement@flag@@
416 \expandafter\xdef\csname opt@\reserved@a\endcsname{%
417 \ifundefined{opt@\reserved@a}\empty%
418 {\csname opt@\reserved@a\endcsname,}%
419 \zap@space#2 \empty%
420 }
421 \expandafter\let
422 \csname opt@#3.#1\expandafter\endcsname
423 \csname opt@\reserved@a\endcsname}
424 \EndIncludeInRelease
425
426 \iflatexrelease\IncludeInRelease{0000/00/00}{\@pass@ptions}
427 \iflatexrelease {\@pass@ptions}%
428 \iflatexrelease
429 \iflatexrelease\def\@pass@ptions#1#2#3{%
430 \expandafter\xdef\csname opt@#3.#1\endcsname{%
431 \ifundefined{opt@#3.#1}\empty%
432 {\csname opt@#3.#1\endcsname,}%
433 \zap@space#2 \empty%
434 }
435 \EndIncludeInRelease
436
437 \onlypreamble\@pass@ptions
438
439 \def\PassOptionsToPackage{\@pass@ptions\@pkgextension}
440 \def\PassOptionsToClass{\@pass@ptions\@clsextension}
441 \onlypreamble\PassOptionsToPackage
442 \onlypreamble\PassOptionsToClass

```

*(End definition for \PassOptionsToPackage and \PassOptionsToClass.)*

\DeclareOption Adds an option as a \ds@ command, or the default \default@ds command.  
 \DeclareOption\* <sup>433</sup> \def\DeclareOption{%

```

434 \let\@fileswith@opti@ns\@badrequireerror
435 \@ifstar\@defdefault@ds\@declareoption}
436 \long\def\@declareoption#1#2{%
437 \xdef\@declaredoptions{\@declaredoptions,#1}%
438 \toks@{\#2}%
439 \expandafter\edef\csname ds@#1\endcsname{\the\toks@}%
440 \long\def\@defdefault@ds#1{%
441 \toks@{\#1}%
442 \edef\default@ds{\the\toks@}%
443 \onlypreamble\DeclareOption
444 \onlypreamble\@declareoption
445 \onlypreamble\@defdefault@ds

```

(End definition for `\DeclareOption` and `\DeclareOption*`.)

`\OptionNotUsed` If we are in a class file, add `\CurrentOption` to the list of unused options. Otherwise, in a package file do nothing.

```

446 </2ekernel>
447 <latexrelease>\IncludeInRelease{2021/06/01}%
448 <latexrelease> {\OptionNotUsed}{filter unused option list}%
449 <*2ekernel | latexrelease>
450 \def\@remove@eq@value#1=#2\@nil{#1}

451 \def\OptionNotUsed{%
452 \ifx\@currext\@clsextension
453 \xdef\@unusedoptionlist{%
454 \ifx\@unusedoptionlist\@empty\else\@unusedoptionlist,\fi
455 \expandafter\@remove@eq@value\CurrentOption=\@nil}%
456 \fi}
457 </2ekernel | latexrelease>
458 <latexrelease>\EndIncludeInRelease
459 <latexrelease>\IncludeInRelease{0000/00/00}%
460 <latexrelease> {\OptionNotUsed}{filter unused option list}%
461 <latexrelease>\let\@remove@eq@value\@undefined

462 <latexrelease>\def\OptionNotUsed{%
463 <latexrelease> \ifx\@currext\@clsextension
464 <latexrelease> \xdef\@unusedoptionlist{%
465 <latexrelease> \ifx\@unusedoptionlist\@empty\else\@unusedoptionlist,\fi
466 <latexrelease> \CurrentOption}%
467 <latexrelease> \fi}
468 <latexrelease>\EndIncludeInRelease
469 <*2ekernel>

470 \onlypreamble\OptionNotUsed

```

(End definition for `\OptionNotUsed` and `\@remove@eq@value`.)

`\default@ds` The default option code. Set by `\@onefilewithoptions` to either `\OptionNotUsed` for classes, or `\@unknownoptionerror` for packages. This may be reset in either case with `\DeclareOption*`.

```
471 % \let\default@ds\OptionNotUsed
```

(End definition for `\default@ds`.)

\ProcessOptions \ProcessOptions calls \ds@option for each known package option, then calls \default@ds for each option on the local options list. Finally resets all the declared options to \relax. The empty option does nothing, this has to be reset on the off chance it's set to \relax if an empty element gets into the \@declaredoptions list.

The star form is similar but executes options given in the order specified in the document, not the order they are declared in the file. In the case of packages, global options are executed before local ones.

```

472 \def\ProcessOptions{%
473 \let\ds@\empty
474 \edef\curroptions{\@optionlist{\currname.\@currext}{}%
475 \@ifstar\xprocessoptions\processoptions}
476 \onlypreamble\ProcessOptions

477 \def\processoptions{%
478 \@for\CurrentOption:=\@declaredoptions\do{%
479 \ifx\CurrentOption\empty\else
480 \expandafter\in@\CurrentOption,\@currext\}%
481 ,\ifx\@currext\clsextension\else\classoptionslist,\fi
482 \curroptions,\}%
483 \ifin@
484 \useoption
485 \expandafter\let\csname ds@\CurrentOption\endcsname\empty
486 \fi
487 }%
488 \processoptions
489 \onlypreamble\processoptions

490 </2ekernel>
491 <latexrelease>\IncludeInRelease{2021/06/01}%
492 <latexrelease> {\@xprocessoptions}{safer @xprocessoptions}%
493 <2ekernel | latexrelease>
494 \def\xprocessoptions{%
495 \ifx\@currext\clsextension\else
496 \ifx\classoptionslist\relax\else
497 \@for\CurrentOption:=\classoptionslist\do{%
498 \ifx\CurrentOption\empty\else
499 \ifundefined{ds@\CurrentOption}{}{%
500 \useoption
501 \expandafter\let\csname ds@\CurrentOption\endcsname\empty
502 }%
503 }%
504 }%
505 \fi
506 \processoptions
507 </2ekernel | latexrelease>
508 <latexrelease>\EndIncludeInRelease
509 <latexrelease>\IncludeInRelease{0000/00/00}%
510 <latexrelease> {\@xprocessoptions}{safer @xprocessoptions}%
511 <latexrelease>\let\@remove@eq@value\undefined

512 <latexrelease>\def\xprocessoptions{%
513 <latexrelease> \ifx\@currext\clsextension\else
514 <latexrelease> \@for\CurrentOption:=\classoptionslist\do{%
515 <latexrelease> \ifx\CurrentOption\empty\else

```

```

516 〈\latexrelease〉 \@expandtwoargs\in@{\, \CurrentOption,\}{}{\@declaredoptions,\}%
517 〈\latexrelease〉 \ifin@
518 〈\latexrelease〉 \use@option
519 〈\latexrelease〉 \expandafter\let\csname ds@\CurrentOption\endcsname\empty
520 〈\latexrelease〉 \fi
521 〈\latexrelease〉 \fi}%
522 〈\latexrelease〉 \fi
523 〈\latexrelease〉 \@process@pti@ns}
524 〈\latexrelease〉\EndIncludeInRelease
525 〈*2ekernel〉

526 \onlypreamble\@process@ptions
```

The common part of \ProcessOptions and \ProcessOptions\*.

```

527 〈/2ekernel〉
528 〈*2ekernel | \latexrelease〉
529 〈\latexrelease〉\IncludeInRelease{2020/10/01}%
530 〈\latexrelease〉 {\@process@pti@ns}{Unused options issue}%
531 〈\def\@process@pti@ns\%}
532 〈\for\CurrentOption:=\@curroptions\do\%
533 〈\ifundefined{ds@\CurrentOption}\%
534 〈\use@option
535 〈\default@ds\%}
```

There should not be any non-empty definition of \CurrentOption at this point, as all the declared options were executed earlier. This is for compatibility with 2.09 styles which use \def\ds@... directly, and so have options which do not appear in \@declaredoptions.

```
536 〈\use@option\%
```

Clear all the definitions for option code. First set all the declared options to \relax, then reset the ‘default’ and ‘empty’ options. and the list of declared options.

```

537 〈\for\CurrentOption:=\@declaredoptions\do\%
538 〈\expandafter\let\csname ds@\CurrentOption\endcsname\relax\%}

539 〈\let\CurrentOption\empty
540 〈\let\@fileswith@pti@ns\@fileswith@pti@ns
541 〈\AtEndOfPackage{〈\expandafter\let
542 〈\csname unprocessedoptions-\@currname.\@currext\endcsname
543 〈\relax\}}
544 〈\onlypreamble\@process@pti@ns
545 〈/2ekernel | \latexrelease〉
546 〈\latexrelease〉\EndIncludeInRelease
547 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
548 〈\latexrelease〉 {\@process@pti@ns}{Unused options issue}%
549 〈\latexrelease〉
550 〈\latexrelease〉\def\@process@pti@ns\%
551 〈\latexrelease〉 〈\for\CurrentOption:=\@curroptions\do\%
552 〈\latexrelease〉 \ifundefined{ds@\CurrentOption}\%
553 〈\latexrelease〉 \use@option
554 〈\latexrelease〉 \default@ds\%
555 〈\latexrelease〉 \use@option\%
556 〈\latexrelease〉 〈\for\CurrentOption:=\@declaredoptions\do\%
557 〈\latexrelease〉 \expandafter\let\csname ds@\CurrentOption\endcsname\relax\%
558 〈\latexrelease〉 〈\let\CurrentOption\empty
559 〈\latexrelease〉 〈\let\@fileswith@pti@ns\@fileswith@pti@ns
560 〈\latexrelease〉 〈\AtEndOfPackage{〈\let\unprocessedoptions\relax\}}
```

```

561 〈\latexrelease〉\EndIncludeInRelease
562 〈*2ekernel〉

(End definition for \ProcessOptions and \ProcessOptions*.)
```

\@options \@options is a synonym for \ProcessOptions\* for upward compatibility with L<sup>A</sup>T<sub>E</sub>X2.09 style files.

```

563 \def\@options{\ProcessOptions*}
564 \@onlypreamble\@options
```

(End definition for \@options.)

\@use@option Execute the code for the current option.

```

565 〈/2ekernel〉
566 〈\latexrelease〉\IncludeInRelease{2021/06/01}%
567 〈\latexrelease〉 {\@use@option}{filter unused option list}%
568 〈*2ekernel | latexrelease〉
569 \def\@use@option{%
570 \@expandtwoargs\@removeelement
571 {\expandafter\@remove@eq@value\CurrentOption=\@nil}%
572 \unusedoptionlist\@unusedoptionlist
573 \csname ds@\CurrentOption\endcsname}
574 〈/2ekernel | latexrelease〉
575 〈\latexrelease〉\EndIncludeInRelease
576 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
577 〈\latexrelease〉 {\@use@option}{filter unused option list}%
578 \def\@use@option{%
579 〈\latexrelease〉 \@expandtwoargs\@removeelement\CurrentOption
580 〈\latexrelease〉 \unusedoptionlist\@unusedoptionlist
581 〈\latexrelease〉 \csname ds@\CurrentOption\endcsname}
582 〈\latexrelease〉\EndIncludeInRelease
583 〈*2ekernel〉
584 \@onlypreamble\@use@option
```

(End definition for \@use@option.)

\ExecuteOptions \ExecuteOptions{\{option-list\}} executes the code declared for each option.

```

585 〈/2ekernel〉
586 〈\latexrelease〉\IncludeInRelease{2017/01/01}%
587 〈\latexrelease〉 {\ExecuteOptions}{Spaces in \ExecuteOptions}%
588 〈*2ekernel | latexrelease〉
589 \def\ExecuteOptions#1{%
```

Use \@fortmp here as it is anyway cleared during \@for loop so does not change any existing names.

```

590 \edef\@fortmp{\zap@space#1 \@empty}%
591 \def\reserved@a##1\@nil{%
592 \@for\CurrentOption:=\@fortmp\do
593 {\csname ds@\CurrentOption\endcsname}%
594 \edef\CurrentOption{\##1}%
595 \expandafter\reserved@a\CurrentOption\@nil}
596 〈/2ekernel | latexrelease〉
597 〈\latexrelease〉\EndIncludeInRelease
598 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
```

```

599 〈latexrelease〉 {\ExecuteOptions}{Spaces in \ExecuteOptions}%
600 〈latexrelease〉\def\ExecuteOptions#1{%
601 〈latexrelease〉 \def\reserved@a##1\@nil{%
602 〈latexrelease〉 \@for\CurrentOption:=#1\do
603 〈latexrelease〉 {\csname ds@\CurrentOption\endcsname}%
604 〈latexrelease〉 \edef\CurrentOption{##1}%
605 〈latexrelease〉 \expandafter\reserved@a\CurrentOption\@nil}%
606 〈latexrelease〉\EndIncludeInRelease
607 {*2ekernel}
608 \Onlypreamble\ExecuteOptions

```

(*End definition for \ExecuteOptions.*)

The top-level commands, which just set some parameters then call the internal command, \@fileswithoptions.

**\documentclass** The main new-style class declaration.

```

609 \def\documentclass{%
610 \let\documentclass@twoclasseserror
611 \if@compatibility\else\let\usepackage\RequirePackage\fi
612 \@fileswithoptions\@clsextension}
613 \Onlypreamble\documentclass

```

(*End definition for \documentclass.*)

**\documentstyle** 2.09 style class ‘style’ declaration.

```

614 \def\documentstyle{%
615 \makeatletter\input{latex209.def}\makeatother
616 \documentclass}
617 \Onlypreamble\documentstyle

```

(*End definition for \documentstyle.*)

**\RequirePackage** Load package if not already loaded.

```

618 \def\RequirePackage{%
619 \@fileswithoptions\@pkgextension}
620 \Onlypreamble\RequirePackage

```

(*End definition for \RequirePackage.*)

**\LoadClass** Load class.

```

621 \def\LoadClass{%
622 \ifx\@currext\@pkgextension
623 \@latex@error
624 {\noexpand\LoadClass in package file}%
625 {You may only use \noexpand\LoadClass in a class file.}%
626 \fi
627 \@fileswithoptions\@clsextension}
628 \Onlypreamble\LoadClass

```

(*End definition for \LoadClass.*)

\@loadwithoptions Pass the current option list on to a class or package. #1 is \cls-or-pkgextension, #2 is \RequirePackage or \LoadClass, #3 is the class or package to be loaded.

```

629 〈/2ekernel〉
630 〈latexrelease〉\IncludeInRelease{2021/06/01}%
631 〈latexrelease〉 {\@loadwithoptions}{Raw option lists load with options}%
632 〈*2ekernel | latexrelease〉
633 \def\@loadwithoptions#1#2#3{%
634 \expandafter\let\csname opt@#3.#1\expandafter\endcsname
635 \csname opt@\currname.\currname\endcsname
636 \expandafter\let\csname raw@opt@#3.#1\expandafter\endcsname
637 \csname raw@opt@\currname.\currname\endcsname
638 #2{#3}%
639 〈/2ekernel | latexrelease〉
640 〈latexrelease〉\EndIncludeInRelease
641 〈latexrelease〉\IncludeInRelease{0000/00/00}
642 〈latexrelease〉 {\@loadwithoptions}{Raw option lists load with options}%
643 〈latexrelease〉\def\@loadwithoptions#1#2#3{%
644 〈latexrelease〉 \expandafter\let\csname opt@#3.#1\expandafter\endcsname
645 〈latexrelease〉 \csname opt@\currname.\currname\endcsname
646 〈latexrelease〉 #2{#3}%
647 〈latexrelease〉\EndIncludeInRelease
648 〈*2ekernel〉
649 \onlypreamble\@loadwithoptions

```

(*End definition for \@loadwithoptions.*)

\LoadClassWithOptions Load class ‘#1’ with the current option list.

```

650 \def\LoadClassWithOptions{%
651 \@loadwithoptions\@clsextension\LoadClass}
652 \onlypreamble\LoadClassWithOptions

```

(*End definition for \LoadClassWithOptions.*)

\RequirePackageWithOptions Load package ‘#1’ with the current option list.

```

653 〈/2ekernel〉
654 〈*2ekernel | latexrelease〉
655 〈latexrelease〉\IncludeInRelease{2020/10/01}%
656 〈latexrelease〉 {\RequirePackageWithOptions}{Unused options issue}%
657 \def\RequirePackageWithOptions{%

```

The resetting of the unprocessed options is now done on a per package basis.

```

658 \AtEndOfPackage{\expandafter\let
659 \csname unprocessedoptions-\currname.\currname\endcsname
660 \relax}%
661 \@loadwithoptions\@pkextension\RequirePackage}
662 \onlypreamble\RequirePackageWithOptions
663 〈/2ekernel | latexrelease〉
664 〈latexrelease〉\EndIncludeInRelease
665 〈latexrelease〉\IncludeInRelease{0000/00/00}%
666 〈latexrelease〉 {\RequirePackageWithOptions}{Unused options issue}%
667 〈latexrelease〉
668 〈latexrelease〉\def\RequirePackageWithOptions{%
669 〈latexrelease〉 \AtEndOfPackage{\let\unprocessedoptions\relax}%

```

```

670 〈\latexrelease〉 \@loadwithoptions\@pkgextension\RequirePackage}
671 〈\latexrelease〉\EndIncludeInRelease
672 〈*2ekernel〉

```

(End definition for `\RequirePackageWithOptions.`)

`\usepackage` To begin with, `\usepackage` produces an error. This is reset by `\documentclass`.

```

673 \def\usepackage#1{%
674 \@latex@error
675 {\noexpand \usepackage before \string\documentclass}%
676 {\noexpand \usepackage may only appear in the document
677 preamble, i.e.,\MessageBreak
678 between \noexpand\documentclass and
679 \string\begin{document}.}%
680 \@gobble
681 \onlypreamble\usepackage

```

(End definition for `\usepackage.`)

`\NeedsTeXFormat` Check that the document is running on the correct system.

```

682 \def\NeedsTeXFormat#1{%
683 \def\reserved@a{#1}%
684 \ifx\reserved@a\fmtname
685 \expandafter\@needsformat
686 \else
687 \@latex@error{This file needs format ‘\reserved@a’}%
688 \MessageBreak but this is ‘\fmtname’}{%
689 The current input file will not be processed
690 further,\MessageBreak
691 because it was written for some other flavor of
692 TeX.\MessageBreak\@ehd}%

```

If the file is not meant to be processed by L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> we stop inputting it, but we do not end the run. We just end inputting the current file.

```

693 \endinput \fi}
694 \onlypreamble\NeedsTeXFormat
695 \def\@needsformat{%
696 \@ifnextchar[%]
697 \@needsf@rmat
698 {}}
699 \onlypreamble\@needsformat
700 \def\@needsf@rmat[#1]{%
701 \@ifl@t@r\fmtversion{#1}{}{%
702 \only@warning@no@line
703 {You have requested release ‘#1’ of LATEX,\MessageBreak
704 but only release ‘\fmtversion’ is available}}}
705 \onlypreamble\@needsf@rmat

```

(End definition for `\NeedsTeXFormat.`)

`\zap@space` `\zap@space foo<space>\@empty` removes all spaces from `foo` that are not protected by `{ }` groups.

```

706 \def\zap@space#1 #2{%
707 #1%

```

```

708 \ifx#2\@empty\else\expandafter\zap@space\fi
709 #2}

```

(End definition for \zap@space.)

\@fileswithoptions The common part of \documentclass and \usepackage.

```

710 \def\@fileswithoptions#1{%
711 \@ifnextchar[%]
712 {\@fileswithoptions#1%}
713 {\@fileswithoptions#1[]}}
714 \onlypreamble\@fileswithoptions

715 \def\@fileswithoptions#1[#2]#3{%
716 \ifnextchar[%]
717 {\@fileswithoptions#1[#2]#3%}
718 {\@fileswithoptions#1[#2]#3[]}}
719 \onlypreamble\@fileswithoptions

```

Then we do some work.

First of all, we define the global variables. Then we look to see if the file has already been loaded. If it has, we check that it was first loaded with at least the current options. If it has not, we add the current options to the package options, set the default version to be 0000/00/00, and load the file if we can find it. Then we check the version number.

Finally, we restore the old file name, reset the default option, and we set the catcode of @.

For classes, we can immediately process the file. For other types, #2 could be a comma separated list, so loop through, processing each one separately.

```

720 </2ekernel>
721 <latexrelease>\IncludeInRelease{2020/10/01}%
722 <latexrelease> {\@fileswithoptions}{\ifx tests in \@fileswithoptions}{%
723 <*2ekernel | latexrelease>
724 \def\@fileswithoptions#1[#2]#3[#4]{%
725 \ifx#1\@clsextension
726 \ifx\@classoptionslist\relax
727 \xdef\@classoptionslist{\zap@space#2 \@empty}%

```

Save raw class list.

```

728 \gdef\@raw@classoptionslist{#2}%
729 \def\reserved@a{%
730 \onefilewithoptions{#3}{#2}{#4}{#1}%
731 \documentclasshook}%
732 \else
733 \def\reserved@a{%
734 \onefilewithoptions{#3}{#2}{#4}{#1}%
735 \fi
736 \else

```

build up a list of calls to \onefilewithoptions (one for each package) without thrashing the parameter stack.

```

737 \def\reserved@b##1,{%

```

If #1 is \onn nil we have reached the end of the list (older version used \nil here but \nil is undefined so \ifx equal to all undefined commands)

```

738 \ifx\@nnil##1\relax\else

```

If `\ifx\@nnil##1\@nnil` is true then #1 is (presumably) empty (Older code used `\relax` which is slightly easier to get into #1 by mistake, which would spoil this test.)

```

739 \ifx\@nnil##1\@nnil\else
740 \noexpand\@onefilewithoptions##1[{\unexpanded{#2}}][{#4}]%
741 \noexpand\@pkgextension
742 \fi
743 \expandafter\reserved@b
744 \fi}%
745 \edef\reserved@a{\zap@space#3 \empty}%
746 \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nnil,}%
747 \fi
748 \reserved@a}
749 //2ekernel | latexrelease)
750 <latexrelease>\EndIncludeInRelease
751 <latexrelease>\IncludeInRelease{2017/01/01}%
752 <latexrelease> {\@fileswith@pti@ns}{ifx tests in \@fileswith@pti@ns}%
753 <latexrelease>\def\@fileswith@pti@ns#1[#2]#3[#4]{%
754 <latexrelease> \ifx#1\@clsextension
755 <latexrelease> \ifx\@classoptionslist\relax
756 <latexrelease> \xdef\@classoptionslist{\zap@space#2 \empty}%
757 <latexrelease> \def\reserved@a{%
758 <latexrelease> \@onefilewithoptions#3[{#2}][{#4}]#1%
759 <latexrelease> \@documentclasshook}%
760 <latexrelease> \else
761 <latexrelease> \def\reserved@a{%
762 <latexrelease> \@onefilewithoptions#3[{#2}][{#4}]#1}%
763 <latexrelease> \fi
764 <latexrelease> \else
765 <latexrelease> \def\reserved@b##1{%
766 <latexrelease> \ifx\@nnil##1\relax\else
767 <latexrelease> \ifx\@nnil##1\@nnil\else
768 <latexrelease> \noexpand\@onefilewithoptions##1[{#2}][{#4}]%
769 <latexrelease> \noexpand\@pkgextension
770 <latexrelease> \fi
771 <latexrelease> \expandafter\reserved@b
772 <latexrelease> \fi}%
773 <latexrelease> \edef\reserved@a{\zap@space#3 \empty}%
774 <latexrelease> \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nnil,}%
775 <latexrelease> \fi
776 <latexrelease> \reserved@a}
777 <latexrelease>\EndIncludeInRelease
778 <latexrelease>\IncludeInRelease{0000/00/00}%
779 <latexrelease> {\@fileswith@pti@ns}{ifx tests in \@fileswith@pti@ns}%
780 <latexrelease>\def\@fileswith@pti@ns#1[#2]#3[#4]{%
781 <latexrelease> \ifx#1\@clsextension
782 <latexrelease> \ifx\@classoptionslist\relax
783 <latexrelease> \xdef\@classoptionslist{\zap@space#2 \empty}%
784 <latexrelease> \def\reserved@a{%
785 <latexrelease> \@onefilewithoptions#3[{#2}][{#4}]#1%
786 <latexrelease> \@documentclasshook}%
787 <latexrelease> \else
788 <latexrelease> \def\reserved@a{%
789 <latexrelease> \@onefilewithoptions#3[{#2}][{#4}]#1}%

```

```

790 <|latexrelease> \fi
791 <|latexrelease> \else
792 <|latexrelease> \def\reserved@b##1,{%
793 <|latexrelease> \ifx\@nil##1\relax\else
794 <|latexrelease> \ifx\relax##1\relax\else
795 <|latexrelease> \noexpand\@onefilewithoptions##1[##2][##4]%
796 <|latexrelease> \noexpand\@pkgextension
797 <|latexrelease> \fi
798 <|latexrelease> \expandafter\reserved@b
799 <|latexrelease> \fi}%
800 <|latexrelease> \edef\reserved@a{\zap@space#3 \@empty}%
801 <|latexrelease> \edef\reserved@a{%
802 <|latexrelease> \expandafter\reserved@b\reserved@a,\@nil,}%
803 <|latexrelease> \fi
804 <|latexrelease> \reserved@a
805 <|latexrelease>\EndIncludeInRelease
806 <|2ekernel>
807 \onlypreamble\@files@with@pti@ns

```

This macro is used when loading packages or classes.

Have the main argument as #1, so we only need one `\expandafter` above.

```

808 </|2ekernel>
809 <*|2ekernel | latexrelease>
810 <|latexrelease>\IncludeInRelease{2020/10/01}%
811 <|latexrelease> {\@onefilewithoptions}{Hooks and unused options issue}%
812 \def\@onefilewithoptions#1[#2][#3]#4{%

```

We have to sanitise file names, so that something like

```

\usepackage{some/local/path/array}
\usepackage{array}

```

won't load `array.sty` twice. It is remotely possible that those are two different files, but as a matter of principles, we will consider that the base file name uniquely identifies a package, regardless of where it lives. This assumption already holds for file hooks, for example, which address the hook to a file by its base name only.

We'll use `\@expl@@@filehook@set@curr@file@nNN` to parse the file name and return the `<path>` and `<base+ext>` in separate token lists. Further ahead, most operations use `\@currname` which doesn't have a path attached to it; only few actions prepend `\@currpath` to `\@currname` (namely loading, as we have to respect the given path).

A file substituition isn't followed just yet because at this point we are parsing user input, so the file is still what the user asked for, and not the file actually loaded.

```

813 \@expl@@@filehook@set@curr@file@nNN{#1.#4}\reserved@a\reserved@b
814 \edef\reserved@c{\def\noexpand\reserved@c####1%
815 \detokenize\expandafter{\expanded{.##4}}%
816 \noexpand\@nil{\def\noexpand\reserved@a{####1}}}\reserved@c
817 \expandafter\reserved@c\reserved@a\@nil
818 \pushfilename
819 \xdef\@currname{\string@makeletter\reserved@a}%
820 \xdef\@currpath{\ifx\reserved@b\@empty\else\reserved@b/\fi}%
821 \global\let\@currext\@currpath

```

The command `\ver@⟨file⟩.⟨ext⟩` is used to signal that a package is already loaded, either because it is in fact loaded, or because its loading was suppressed. In minimal installations, said package may not exist but still have its loading suppressed with `\ver@⟨file⟩.⟨ext⟩`, so before checking if the file exists we have to check that we do need to load it with `\@ifl@aded`. If we don't, then there's no point in checking for a typo or load-disabling.

```
822 \@ifl@aded\@currext\@currname
```

In the current preferred approach, a key family name will exist for processing using `\tkeys`. In that case, we replace the previous package options with the new ones, then call the key handler. Otherwise, we use the more classical clash handler.

```
823 {%
824 \@ifundefined{opt@fam@\@currname.\@currext}%
825 {\@onefilewithoptions@clashchk{\#2}}%
826 {%
827 \@namedef{opt@\@currname.\@currext}{\#2}%
828 \expandafter\expandafter\expandafter\ProcessKeyPackageOptions
829 \expandafter\expandafter\expandafter\expandafter
830 [\csname opt@fam@\@currname.\@currext\endcsname]%
831 }%
832 }%
833 {\makeatletter
```

The next line seems to be necessary for 2.09 compatibility (the way the code is written there) This seems questionable and should be looked at as in 2e it is definitely unnecessary at this point!

```
834 \@reset@ptions
```

First we take the `⟨name⟩` and `⟨ext⟩` given in the argument and check if the file exists, and issue an error otherwise asking for a correction with `\@missingfileerror`. For checking if the file exists we use `\@currpath` (usually empty) before `\@currname`.

```
835 \IfFileExists{\@currpath\@currname.\@currext}{}{%
836 {\@missing@onefilewithoptions{\#2}}%
```

If `\@currname` is empty (the user replied to the “Enter file name” prompt with `<RETURN>`), so stop here (do `\@popfilename` to pop the item just added above).

This `\@gobble` omits the date check at the end.

```
837 \ifx\@currname\@empty
838 \expandafter\@gobble
839 \else
```

If the file exists, check if it was load-prevented, and otherwise do the bookkeeping with `\@filehook@file@push` then call `\set@curr@file` to set `\@curr@file` (and do any required substitution), then actually load the class/package with `\load@onefile@withoptions`. `\set@curr@file` also needs the file path.

```
840 \@disable@packageload@do{\@currname.\@currext}%
841 {\@expl@@@filehook@file@push@@
842 \set@curr@file{\@currpath\@currname.\@currext}%
843 \@filehook@set@CurrentFile}
```

The `\set@curr@file` line above might have replaced the file, so `\@currname` and `\@currext` may no longer hold the actual package being loaded, so in that case we need to update these two token lists (`\@curr@file` holds the file name after replacement, so we parse that).

The requested file is saved in `\@currpkg@reqd` to be used in `\InputIfFileExists` later: if the updated `\@currname` and `\@currext` are used we lose track of the substitution, so `\CurrentFile` and `\CurrentFileUsed` will be (incorrectly) the same.

```

844 \expandafter\@swaptwoargs\expandafter
845 {\expandafter{\@currpkg@reqd}}%
846 {%
847 <
848 \edef\@currpkg@reqd{\@currname.\@currext}%
849 \ifx\CurrentFile\CurrentFileUsed
850 \else
851 \filename@parse\@curr@file
852 \edef\@currpath{\string@makeletter\filename@area}%
853 \edef\@currname{\string@makeletter\filename@base}%
854 \edef\@currext{\string@makeletter\filename@ext}%
855 \fi
856 \load@onefile@withoptions{#2}%
857 \def\@currpkg@reqd{\@currpkg@reqd}%
858 }%
859 }%
860 >

```

Now just clean up and exit.

```

858 \expl@@@filehook@file@pop@@}%
859 \expandafter\@firstofone
860 \fi}%

```

Except in the case where `\@currname` is empty, the date is checked against the date marked in the package file:

```

861 {\@ifl@ter\@currext{\@currname}{#3}{}}%
862 {\@latex@warning@no@line
863 {You have requested,\on@line,
864 version\MessageBreak
865 '#3' of \cls@pkg\space \@currname,\MessageBreak
866 but only version\MessageBreak
867 '\csname ver@\@currname.\@currext\endcsname'\MessageBreak
868 is available}}%
869
870 \ifx\@currext@\clsextension\let\LoadClass@twoloadclasserror\fi}%
871 \popfilename
872 \reset@options

```

If the package is already loaded, check that there were no option clashes.

```

872 \def\@onefilewithoptions@clashchk#1{%
873 \@if@ptions\@currext{\@currname}{#1}{}}%
874 {\@latex@error
875 {Option clash for \cls@pkg\space \@currname}%
876 {The package \@currname\space has already been loaded
877 with options:\MessageBreak
878 \space\space[\optionlist{\@currname.\@currext}]\MessageBreak
879 There has now been an attempt to load it
880 with options\MessageBreak
881 \space\space[#1]\MessageBreak
882 Adding the global options:\MessageBreak
883 \space\space

```

```

884 \optionlist{\@currname.\@currext},#1\MessageBreak
885 to your \noexpand\documentclass declaration may fix this.%

886 \MessageBreak
887 Try typing \space <return> \space to proceed.}%
888 \@firstofone}

889 \let\@currpkg@reqd\@empty
890 \onlypreamble\onefilewithoptions
 The kernel no longer uses \unprocessedoptions
891 \let\@unprocessedoptions\@undefined

```

Now the action taken when a file is not found. Path must be included here as it eventually leads to a file lookup.

```

892 \def\@missing@onefilewithoptions#1{%
893 \@missingfileerror{\@currpath\@currname}\@currext
894 \global\let\@currpath\@missingfile@area
895 \global\let\@currname\@missingfile@base
896 \global\let\@currext\@missingfile@ext}

```

Now the code that actually does the file loading:

```

\load@onefile@withoptions 897 \def\load@onefile@withoptions#1{%
898 \let\CurrentOption\@empty
899 \reset@options

```

Grab everything in a macro, so the parameter stack is popped before any processing begins.

```

900 \def\reserved@a{%
901 \pass@options\@currext{#1}{\@currname}%
902 \expandafter\let
903 \csname opt@\@currpkg@reqd\expandafter\endcsname
904 \csname opt@\@currname.\@currext\endcsname
905 \expandafter\let
906 \csname @raw@opt@\@currpkg@reqd\expandafter\endcsname
907 \csname @raw@opt@\@currname.\@currext\endcsname
908 \global\expandafter
909 \let\csname ver0@\@currname.\@currext\endcsname\@empty

```

We initialize \....-h@k here and only if we load the file so that it remains undefined otherwise.

```

910 \expandafter\let\csname\@currname.\@currext-h@k\endcsname\@empty

```

When the current extension is \@pkgeextension we are loading a package otherwise, if it is \@clsextension, a class, so depending on that we execute different hooks. If the extension is neither, then it is another type of file without special hooks.

```

911 %-----%
912 \ifx\@currext\@pkgeextension
913 \UseHook{package/before}%
914 \UseOneTimeHook{package/\@currname/before}%
915 \else
916 \ifx\@currext\@clsextension
917 \UseHook{class/before}%
918 \UseOneTimeHook{class/\@currname/before}%
919 \fi
920 \fi

```

Now actually load the file (at this point we are certain it exists, but use `\InputIfFileExists` so that file hooks are executed). `\@currpath` is needed here too.

```
921 \InputIfFileExists{\@currpath\@currpkg@reqd}{}%
922 {\@latex@error
923 {The \@cls@pkg\space\@currpkg@reqd\space failed to load}\@ehd}%
924 %-----
```

In older versions of the code `\@unprocessedoptions` would generate an error for each specified option in a package unless a `\ProcessOptions` has appeared in the package file.

This has changed in 2020. We now use a separate macro per package to avoid interference in case of nested packages. The whole code for handling this issue (GitHub 22) was provided by Hironobu Yamashita, thanks for that.

```
925 \expandafter\let\csname unprocessedoptions-\@currname.\@currext\endcsname
926 \@@unprocessedoptions
927 \csname\@currname.\@currext-h@k\endcsname
928 \expandafter\let\csname\@currname.\@currext-h@k\endcsname
929 \@undefined
```

Catch the case where the packages has handled the options and redefined `\@unprocessedoptions` to `\relax` (old interface). In that case no error should be produced.

```
930 \ifx\@unprocessedoptions\relax
931 \let\@unprocessedoptions\@undefined
```

Otherwise run the per package set of unused options.

```
932 \else
933 \csname unprocessedoptions-\@currname.\@currext\endcsname
934 \fi
```

In either case we drop the macro afterwards as it is no longer needed.

```
935 \expandafter\let
936 \csname unprocessedoptions-\@currname.\@currext\endcsname
937 \@undefined
```

And same procedure, James, when we are finished loading, except that the hook order is now reversed.

```
938 %-----
939 \ifx\@currext\@pkgextension
940 \UseOneTimeHook{package/\@currname/after}%
941 \UseHook{package/after}%
942 \else
943 \ifx\@currext\@clsextension
944 \UseOneTimeHook{class/\@currname/after}%
945 \UseHook{class/after}%
946 \fi
947 \fi}%
948 %-----
949 \@ifl@aded\@currext\@currname{}{\reserved@a}
```

Now declare the non-generic package and class hooks used above:

```
950 \NewHook{package/before}
951 \NewHook{class/before}
952 \NewReversedHook{package/after}
953 \NewReversedHook{class/after}
```

```

954 {/2ekernel | latexrelease}
955 \end{IncludeInRelease}
956 \IncludeInRelease{0000/00/00}%
957 { \onefilewithoptions}{Hooks and unused options issue}%
958 \end{IncludeInRelease}

```

Because of the way `\@onefilewithoptions` is changed for rollback handling below we have to define `\load@onefilewithoptions` when rolling back!

```

959 \def\load@onefilewithoptions#1[#2][#3]{%
960 \pushfilename
961 \xdef\currname{#1}%
962 \global\let\currext\#4%
963 \let\CurrentOption\empty
964 \resetoptions
965 \makeatletter
966 \def\reserved@a{%
967 \if@aded\currext{#1}%
968 {\@ifoptions{\currext{#1}{#2}{}}{%
969 \@latex@error
970 {Option clash for \cls@pkg\space #1}%
971 {The package #1 has already been loaded
972 with options:\MessageBreak
973 \space\space[\optionlist{#1.\currext}]\MessageBreak
974 There has now been an attempt to load it
975 with options\MessageBreak
976 \space\space[#2]\MessageBreak
977 Adding the global options:\MessageBreak
978 \space\space
979 \optionlist{#1.\currext},#2\MessageBreak
980 to your \noexpand\documentclass declaration may fix this.%\MessageBreak
981 Try typing \space <return> \space to proceed.}}}}%
982 {\@passoptions{\currext{#2}{#1}}%
983 \global\expandafter
984 \let\csname ver@\currname.\currext\endcsname\empty
985 \expandafter\let\csname\currname.\currext-h@k\endcsname\empty
986 \InputIfFileExists
987 {\currname.\currext}%
988 {}%
989 {\@missingfileerror\currname\currext}%
990 \let\unprocessedoptions\@unprocessedoptions
991 \csname\currname.\currext-h@k\endcsname
992 \expandafter\let\csname\currname.\currext-h@k\endcsname
993 \undefined
994 \@unprocessedoptions\%%
995 \if@ter\currext{#1}{#3}{}}%
996 {\@latex@warning@no@line
997 {You have requested,\on@line,
998 version\MessageBreak
999 '#3' of \cls@pkg\space #1,\MessageBreak
1000 but only version\MessageBreak
1001 '\csname ver@\#1.\currext\endcsname'\MessageBreak
1002 is available}}%
1003 \ifx\currext\clsextension\let\LoadClass\@twoloadclasserror\fi

```

```

1005 〈latexrelease〉 \popfilename
1006 〈latexrelease〉 \reset@ptions}%
1007 〈latexrelease〉 \reserved@a}
1008 〈latexrelease〉
1009 〈latexrelease〉\let \load@oneline@withoptions \undefined
1010 〈latexrelease〉\let \missing@oneline@withoptions \undefined
1011 〈latexrelease〉
1012 〈latexrelease〉\EndIncludeInRelease
1013 {*2ekernel}

```

(End definition for `\@files@with@options` and others.)

`\@@files@with@ptions` Save the definition (for error checking).

```

1014 \let\@@files@with@ptions\@files@with@ptions
1015 \onlypreamble\@@files@with@ptions

```

(End definition for `\@@files@with@ptions`.)

`\@reset@ptions` Reset the default option, and clear lists of declared options.

```

1016 \def\@reset@ptions{%
1017 \global\ifx\@currext\@clsextension
1018 \let\default@ds\OptionNotUsed
1019 \else
1020 \let\default@ds\@unknownoptionerror
1021 \fi
1022 \global\let\ds@\emptyset
1023 \global\let\@declaredoptions\emptyset}
1024 \onlypreamble\@reset@ptions

```

(End definition for `\@reset@ptions`.)

## 4.1 Hooks

Allow code to be saved to be executed at specific later times.

Save things in macros, I considered using toks registers, (and `\addto@hook` from the NFSS code, that would require stacking the contents in the case of required packages, so just generate a new macro for each package.

`\@begindocumenthook` Stuff to appear at the beginning or end of the document.

```

1025 \ifx\@begindocumenthook\undefined
1026 \let\@begindocumenthook\emptyset
1027 \fi
1028 \let\@enddocumenthook\emptyset

```

(End definition for `\@begindocumenthook` and `\@enddocumenthook`.)

`\AtEndOfPackage` The access functions.

```

1029 \def\AtEndOfPackage{%
1030 \expandafter\g@addto@macro\csname\@currname.\@currext-h@ok\endcsname}
1031 \let\AtEndOfClass\AtEndOfPackage
1032 \onlypreamble\AtEndOfPackage
1033 \onlypreamble\AtEndOfClass

```

```

1034 </2ekernel>
1035 <*2ekernel | latexrelease>
1036 <latexrelease>\IncludeInRelease{2020/10/01}%
1037 <latexrelease> {\AtBeginDocument}{Use hook system}%
1038 \DeclareRobustCommand\AtBeginDocument{\AddToHook{begindocument}}
1039 \DeclareRobustCommand\AtEndDocument {\AddToHook{enddocument}}
1040 \% \DeclareRobustCommand\AtEndDocument {\AddToHook{env/document/end}} % alternative impl
1041 </2ekernel | latexrelease>
1042 <latexrelease>\EndIncludeInRelease
1043 <latexrelease>\IncludeInRelease{0000/00/00}%
1044 <latexrelease> {\AtBeginDocument}{Use hook system}%
1045 <latexrelease>
1046 <latexrelease>\DeclareRobustCommand\AtBeginDocument{\g@addto@macro\@begindocumenthook}
1047 <latexrelease>\DeclareRobustCommand\AtEndDocument{\g@addto@macro\@enddocumenthook}
1048 <latexrelease>
1049 <latexrelease>\EndIncludeInRelease
1050 <*2ekernel>
1051 \onlypreamble\AtBeginDocument

```

(*End definition for \AtEndOfPackage and others.*)

\@cls@pkg The current file type.

```

1052 \def\@cls@pkg{%
1053 \ifx\@currext\@clsextension
1054 document class%
1055 \else
1056 package%
1057 \fi}
1058 \onlypreamble\@cls@pkg

```

(*End definition for \@cls@pkg.*)

\@unknownoptionerror Bad option.

```

1059 \def\@unknownoptionerror{%
1060 \@latex@error
1061 {Unknown option ‘\CurrentOption’ for \@cls@pkg\space‘\@currname’}%
1062 {The option ‘\CurrentOption’ was not declared in
1063 \@cls@pkg\space‘\@currname’, perhaps you\MessageBreak
1064 misspelled its name.
1065 Try typing \space <return>
1066 \space to proceed.}}
1067 \onlypreamble\@unknownoptionerror

```

(*End definition for \@unknownoptionerror.*)

\@unprocessedoptions Declare an error for each option, unless a \ProcessOptions occurred.

```

1068 \def\@unprocessedoptions{%
1069 \ifx\@currext\@pkgetension
1070 \edef\@curroptions{\@optionlist{\@currname.\@currext}}%
1071 \@for\CurrentOption:=\@curroptions\do{%
1072 \ifx\CurrentOption\@empty\else\@unknownoptionerror\fi}%
1073 \fi}
1074 \onlypreamble\@unprocessedoptions
1075 \onlypreamble\@unprocessedoptions

```

(End definition for \@@unprocessedoptions.)

\@badrequireerror \RequirePackage or \LoadClass occurs in the options section.  
1076 \def\@badrequireerror#1[#2]#3[#4]{%  
1077 \ @latex@error  
1078 {\noexpand\RequirePackage or \noexpand\LoadClass  
1079 in Options Section}%;  
1080 {The \cls@pkg\space '@currname' is defective.\MessageBreak  
1081 It attempts to load '#3' in the options section, i.e.,\MessageBreak  
1082 between \noexpand\DeclareOption and \string\ProcessOptions.};  
1083 \onlypreamble\@badrequireerror

(End definition for \@badrequireerror.)

\@twoloadclasserror Two \LoadClass in a class.

1084 \def\@twoloadclasserror{%;  
1085 \ @latex@error  
1086 {Two \noexpand\LoadClass commands}%;  
1087 {You may only use one \noexpand\LoadClass in a class file}};  
1088 \onlypreamble\@twoloadclasserror

(End definition for \@twoloadclasserror.)

\@twoclasseserror Two \documentclass or \documentstyle.

1089 \def\@twoclasseserror#1#2{%;  
1090 \ @latex@error  
1091 {Two \noexpand\documentclass or \noexpand\documentstyle commands}%;  
1092 {The document may only declare one class.}\@gobble}  
1093 \onlypreamble\@twoclasseserror

(End definition for \@twoclasseserror.)

## 4.2 Providing shipment

\two@digits Prefix a number less than 10 with '0'.

1094 \def\two@digits#1{\ifnum#1<10 0\fi\number#1}

(End definition for \two@digits.)

\filecontents This environment implements inline files. The star-form does not write extra comments  
\endfilecontents into the file.

1095 </2ekernel>  
1096 {\*2ekernel | latexrelease}  
1097 <| latexrelease>\IncludeInRelease{2020/10/01}%;  
1098 <| latexrelease> \filecontents{Define \q@curr@file directly (gh/220)}%;  
1099 %

We use @tempswa to mean no preamble writing and reuse @files w to indicate no overwriting:

1100 \def\filecontents{\@tempswatru e\@filestrue  
1101 \ @ifnextchar[\filecontents@opt\filecontents  
1102 }  
1103 \ @namedef{filecontents\*}{\@tempswafalse\@filestrue  
1104 \ @ifnextchar[\filecontents@opt\filecontents  
1105 }

To handle the optional argument we execute for each option the command `\filec@ntents@OPTION` if it exist or complain about unknown option.

```

1106 \def\filec@ntents@opt[#1]{%
1107 \edef\@fortmp{\zap@space#1 \empty}%
1108 \for\reserved@a:=\@fortmp\do{%
1109 \ifcsname filec@ntents@\reserved@a\endcsname
1110 \csname filec@ntents@\reserved@a\endcsname
1111 \else
1112 \@latex@error{Unknown filecontents option \reserved@a}%
1113 {Valid options are force (or overwrite), nosearch, noheader}%
1114 \fi}%
1115 \filec@ntents
1116 }

```

Option `force` (or `overwrite`) changes the overwriting switch

```

1117 \let\filec@ntents@force\@filesfalse
1118 \let\filec@ntents@overwrite\@filesfalse % alternative name

```

and option `noheader` the preamble switch (which is equivalent to using the star form of the environment).

```
1119 \let\filec@ntents@noheader\@tempswafalse
```

Option `nosearch` only checks the current directory not the whole TeX tree for the existence of the file to write.

```

1120 \def\filec@ntents@nosearch{%
1121 \let\filec@ntents@checkdir\@currdir
1122 \def\filec@ntents@where{in current directory}}

```

By default we search the whole tree:

```

1123 \let\filec@ntents@checkdir\@empty
1124 \def\filec@ntents@where{exists on the system}

1125 \begingroup%
1126 \tempcnta=1
1127 \loop
1128 \catcode\@tempcnta=12 %
1129 \advance\@tempcnta\@ne %
1130 \ifnum\@tempcnta<32 %
1131 \repeat %
1132 \catcode`*=11 %
1133 \catcode`\^M\active%
1134 \catcode`\^L\active\let^\relax%
1135 \catcode`\^I\active%

```

```

1136 \gdef\filec@ntents#1{%
1137 \set@curr@file{\filec@ntents@checkdir#1}%
1138 \edef\q@curr@file{"\curr@file"}%

```

LuaTeX has more writes (and 18 is safe here).

```

1139 \chardef\reserved@c\ifx\directlua\undefined 15 \else 127 \fi%
1140 \openin\@inputcheck\q@curr@file \space %
1141 \ifeof\@inputcheck%
1142 \@latex@note@no@line%
1143 {Writing file '\@currdir\@curr@file'}%

```

```

1144 \ch@ck7\reserved@c\write\relax%
1145 \immediate\openout\reserved@c\q@curr@file\relax%
1146 \else%
1147 \if@filesw%
1148 \@latex@note@no@line%
1149 {File '\@curr@file' already \filec@ntents@where.\MessageBreak%
1150 Not generating it from this source}%
1151 \let\write@gobbletwo%
1152 \let\closeout@gobble%
1153 \else%

```

If we are overwriting, we try to make sure that the user is not by mistake overwriting the input file (`\jobname`). Of course, this only works for input files ending in `.tex`. If a different extension is used there is no way to see that we are overwriting ourselves!

```

1154 \edef\reserved@b{\detokenize\expandafter{\jobname}}%
1155 \ifx\@curr@file\reserved@b%
1156 \Qfilewtrue%
1157 \else%
1158 \edef\reserved@b{\reserved@b\detokenize{.tex}}%
1159 \ifx\@curr@file\reserved@b%
1160 \Qfilewtrue%
1161 \fi%
1162 \fi%

```

We allocate a write channel but we open it only if it is (hopefully) safe. If not opened that means we are going to write on the terminal.

```

1163 \ch@ck7\reserved@c\write\relax%
1164 \if@filesw% % Foul ... trying to overwrite \jobname!
1165 \@latex@error{Trying to overwrite '\jobname.tex'}{You can't %
1166 write to the file you are reading from!\MessageBreak%
1167 Data is written to screen instead.}%
1168 \else%
1169 \@latex@warning@no@line%
1170 {Writing or overwriting file '\@currdir\@curr@file'}%
1171 \immediate\openout\reserved@c\q@curr@file\relax%
1172 \fi%
1173 \fi%
1174 \fi%

```

Closing the `\@inputcheck` is done here to avoid having to do this in each branch.

```

1175 \closein\@inputcheck%
1176 \if@tempswa%
1177 \immediate\write\reserved@c{%
1178 \percentchar\percentchar\space%
1179 \expandafter\gobble\string\LaTeXe file '\@curr@file'^^J%
1180 \percentchar\percentchar\space generated by the %
1181 '\@currenvir' \expandafter\gobblefour\string\newenvironment^^J%
1182 \percentchar\percentchar\space from source '\jobname' on %
1183 \number\year/\two@digits\month/\two@digits\day.^^J%
1184 \percentchar\percentchar}%
1185 \fi%
1186 \let\do\makeother\dospecials%

```

If there are active characters in the upper half (e.g., from `inputenc` there would be confusion so we render everything harmless.

```

1187 \count@ 128\relax%
1188 \loop%
1189 \catcode\count@ 11\relax%
1190 \advance\count@ \@ne%
1191 \ifnum\count@<\@cclvi%
1192 \repeat%

1193 \edef\E{\@backslashchar end\string{\@currenvir\string}}%
1194 \edef\reserved@b{%
1195 \def\noexpand\reserved@b{%
1196 #####1\E#####2\E#####3\relax}%
1197 \reserved@b{%
1198 \ifx\relax##3\relax%

```

There was no `\end{filecontents}`

```

1199 \immediate\write\reserved@c{##1}%
1200 \else%

```

There was a `\end{filecontents}`, so stop this time.

```

1201 \edef^^M{\noexpand\end{\@currenvir}}%
1202 \ifx\relax##1\relax%
1203 \else%

```

Text before the `\end`, write it with a warning.

```

1204 \@latex@warning{Writing text ‘##1’ before %
1205 \string\end{\@currenvir}\MessageBreak
1206 as last line of \@curr@file}%
1207 \immediate\write\reserved@c{##1}%
1208 \fi%
1209 \ifx\relax##2\relax%
1210 \else%

```

Text after the `\end`, ignore it with a warning.

```

1211 \@latex@warning{%
1212 Ignoring text ‘##2’ after \string\end{\@currenvir}}%
1213 \fi%
1214 \fi%
1215 ^^M}%

1216 \catcode‘\^\^L\active%
1217 \let\^\^L\@undefined%
1218 \def^\^L{\expandafter\ifx\csname L\endcsname\relax\fi ^\^J^\^J}%
1219 \catcode‘\^\^I\active%
1220 \let\^\^I\@undefined%
1221 \def^\^I{\expandafter\ifx\csname I\endcsname\relax\fi \space}%
1222 \catcode‘\^\^M\active%
1223 \edef^\^M##1^\^M{%
1224 \noexpand\reserved@b##1\^\^E\relax}%
1225 \endgroup%

1226 </2ekernel | latexrelease>
1227 <| latexrelease|\EndIncludeInRelease
1228 <| latexrelease|\IncludeInRelease[2019/10/01]%
1229 <| latexrelease> {\filecontents}{Spaces in file names + optional arg}%

```

```

1230 \langle latexrelease\rangle
1231 \langle latexrelease\rangle\def\filecontents{\@tempswattrue\@files wtrue
1232 \langle latexrelease\rangle \ @ifnextchar[\filec@ntents@opt\filec@ntents
1233 \langle latexrelease\rangle}
1234 \langle latexrelease\rangle\@namedef{filecontents*}{\@tempswafalse\@files wtrue
1235 \langle latexrelease\rangle \ @ifnextchar[\filec@ntents@opt\filec@ntents
1236 \langle latexrelease\rangle}
1237 \langle latexrelease\rangle\def\filec@ntents@opt[#1]{%
1238 \langle latexrelease\rangle \edef\@fortmp{\zap@space#1 \empty}%
1239 \langle latexrelease\rangle \for\reserved@a:=\@fortmp\do{%
1240 \langle latexrelease\rangle \ifcsname filec@ntents@\reserved@a\endcsname
1241 \langle latexrelease\rangle \csname filec@ntents@\reserved@a\endcsname
1242 \langle latexrelease\rangle \else
1243 \langle latexrelease\rangle \@latex@error{Unknown filecontents option \reserved@a}%
1244 \langle latexrelease\rangle {Valid options are force (or overwrite), nosearch, noheader}%
1245 \langle latexrelease\rangle \fi}%
1246 \langle latexrelease\rangle \filec@ntents
1247 \langle latexrelease\rangle}
1248 \langle latexrelease\rangle\let\filec@ntents@force\@files wfalse
1249 \langle latexrelease\rangle\let\filec@ntents@overwrite\@files wfalse % alternative name
1250 \langle latexrelease\rangle\let\filec@ntents@noheader\@tempswafalse
1251 \langle latexrelease\rangle\def\filec@ntents@nosearch{%
1252 \langle latexrelease\rangle \let\filec@ntents@checkdir\@currdir
1253 \langle latexrelease\rangle \def\filec@ntents@where{in current directory}}
1254 \langle latexrelease\rangle\let\filec@ntents@checkdir\@empty
1255 \langle latexrelease\rangle\def\filec@ntents@where{exists on the system}
1256 \langle latexrelease\rangle\begingroup%
1257 \langle latexrelease\rangle\@tempcnta=1
1258 \langle latexrelease\rangle\loop
1259 \langle latexrelease\rangle \catcode\@tempcnta=12 %
1260 \langle latexrelease\rangle \advance\@tempcnta\@ne %
1261 \langle latexrelease\rangle\ifnum\@tempcnta<32 %
1262 \langle latexrelease\rangle\repeat %
1263 \langle latexrelease\rangle\catcode`*=11 %
1264 \langle latexrelease\rangle\catcode`\^M\active%
1265 \langle latexrelease\rangle\catcode`\^L\active\let\^L\relax%
1266 \langle latexrelease\rangle\catcode`\^I\active%
1267 \langle latexrelease\rangle\gdef\filec@ntents#1{%
1268 \langle latexrelease\rangle \set@curr@file{\filec@ntents@checkdir#1}%
1269 \langle latexrelease\rangle \edef\q@curr@file{\expandafter\quote@name\expandafter{\@curr@file}}%
1270 \langle latexrelease\rangle \chardef\reserved@c\ifx\directlua\@undefined 15 \else 127 \fi%
1271 \langle latexrelease\rangle \openin@\inputcheck\q@curr@file \space %
1272 \langle latexrelease\rangle \ifeof\inputcheck%
1273 \langle latexrelease\rangle \@latex@warning{no@line}%
1274 \langle latexrelease\rangle {Writing file '\@currdir\@curr@file'}%
1275 \langle latexrelease\rangle \ch@ck7\reserved@c\write\relax%
1276 \langle latexrelease\rangle \immediate\openout\reserved@c\q@curr@file\relax%
1277 \langle latexrelease\rangle \else%
1278 \langle latexrelease\rangle \if@files w%
1279 \langle latexrelease\rangle \@latex@warning{no@line}%
1280 \langle latexrelease\rangle {File '\@curr@file' already \filec@ntents@where.\MessageBreak}%
1281 \langle latexrelease\rangle {Not generating it from this source}%
1282 \langle latexrelease\rangle \let\write\@gobbletwo%
1283 \langle latexrelease\rangle \let\closeout\@gobble%

```

```

1284 〈\latexrelease〉 \else%
1285 〈\latexrelease〉 \edef\reserved@a{\#1}%
1286 〈\latexrelease〉 \edef\reserved@a{\detokenize\expandafter{\reserved@a}}%
1287 〈\latexrelease〉 \edef\reserved@b{\detokenize\expandafter{\jobname}}%
1288 〈\latexrelease〉 \ifx\reserved@a\reserved@b%
1289 〈\latexrelease〉 \@fileswtrue%
1290 〈\latexrelease〉 \else%
1291 〈\latexrelease〉 \edef\reserved@b{\reserved@b\detokenize{.tex}}%
1292 〈\latexrelease〉 \ifx\reserved@a\reserved@b%
1293 〈\latexrelease〉 \@fileswtrue%
1294 〈\latexrelease〉 \fi%
1295 〈\latexrelease〉 \fi%
1296 〈\latexrelease〉 \ch@ck7\reserved@c\write\relax%
1297 〈\latexrelease〉 \if@filesw% % Foul ... trying to overwrite \jobname!
1298 〈\latexrelease〉 \@latex@error{Trying to overwrite '\jobname.tex'{You can't %
1299 〈\latexrelease〉 write to the file you are reading from!\MessageBreak%
1300 〈\latexrelease〉 Data is written to screen instead.}%
1301 〈\latexrelease〉 \else%
1302 〈\latexrelease〉 \@latex@warning@no@line%
1303 〈\latexrelease〉 {Writing or overwriting file '@currdir@\curr@file'}%
1304 〈\latexrelease〉 \immediate\openout\reserved@c\q@curr@file\relax%
1305 〈\latexrelease〉 \fi%
1306 〈\latexrelease〉 \fi%
1307 〈\latexrelease〉 \fi%
1308 〈\latexrelease〉 \closein\@inputcheck%
1309 〈\latexrelease〉 \if@tempswa%
1310 〈\latexrelease〉 \immediate\write\reserved@c{%
1311 〈\latexrelease〉 \@percentchar\@percentchar\space%
1312 〈\latexrelease〉 \expandafter\@gobble\string\LaTeXe file '@curr@file'^^J%
1313 〈\latexrelease〉 \@percentchar\@percentchar\space generated by the %
1314 〈\latexrelease〉 '@currenvir' \expandafter\@gobblefour\string\newenvironment^^J%
1315 〈\latexrelease〉 \@percentchar\@percentchar\space from source '\jobname' on %
1316 〈\latexrelease〉 \number\year/\two@digits\month/\two@digits\day.^^J%
1317 〈\latexrelease〉 \@percentchar\@percentchar}%
1318 〈\latexrelease〉 \fi%
1319 〈\latexrelease〉 \let\do\@makeother\dospecials%
1320 〈\latexrelease〉 \count@ 128\relax%
1321 〈\latexrelease〉 \loop%
1322 〈\latexrelease〉 \catcode\count@ 11\relax%
1323 〈\latexrelease〉 \advance\count@ \@ne%
1324 〈\latexrelease〉 \ifnum\count@<\@ccvi%
1325 〈\latexrelease〉 \repeat%
1326 〈\latexrelease〉 \edef\E{\@backslashchar end\string{@currenvir\string}{}%
1327 〈\latexrelease〉 \edef\reserved@b{%
1328 〈\latexrelease〉 \def\noexpand\reserved@b{%
1329 〈\latexrelease〉 #####1\@####2\@####3\relax}%
1330 〈\latexrelease〉 \reserved@b{%
1331 〈\latexrelease〉 \ifx\relax##3\relax%
1332 〈\latexrelease〉 \immediate\write\reserved@c{##1}%
1333 〈\latexrelease〉 \else%
1334 〈\latexrelease〉 \edef\@M{\noexpand\end{@currenvir}}%
1335 〈\latexrelease〉 \ifx\relax##1\relax%
1336 〈\latexrelease〉 \else%
1337 〈\latexrelease〉 \@latex@warning{Writing text '##1' before %

```

```

1338 <|latexrelease> \string\end{\@currenvir}\MessageBreak as last line of \@curr@file}%
1339 <|latexrelease> \immediate\write\reserved@c{##1}%
1340 <|latexrelease> \fi%
1341 <|latexrelease> \ifx\relax##2\relax%
1342 <|latexrelease> \else%
1343 <|latexrelease> \@latex@warning{%
1344 <|latexrelease> Ignoring text '##2' after \string\end{\@currenvir}}%
1345 <|latexrelease> \fi%
1346 <|latexrelease> \fi%
1347 <|latexrelease> ^~M}%
1348 <|latexrelease> \catcode`^~L\active%
1349 <|latexrelease> \let\L\@undefined%
1350 <|latexrelease> \def^~L{\expandafter\ifx\csname L\endcsname\relax\fi ^~J^~J}%
1351 <|latexrelease> \catcode`^~I\active%
1352 <|latexrelease> \let\I\@undefined%
1353 <|latexrelease> \def^~I{\expandafter\ifx\csname I\endcsname\relax\fi\space}%
1354 <|latexrelease> \catcode`^~M\active%
1355 <|latexrelease> \edef^~M##1^~M{%
1356 <|latexrelease> \noexpand\reserved@b##1\relax}%
1357 <|latexrelease>\endgroup%
1358 <|latexrelease>\EndIncludeInRelease
1359 <|latexrelease>\IncludeInRelease{0000/00/00}%
1360 <|latexrelease> {\filec@ntents}{Spaces in file names + optional arg}%
1361 <|latexrelease>
1362 <|latexrelease>\let\filec@ntents@opt \@undefined
1363 <|latexrelease>\let\filec@ntents@force \@undefined
1364 <|latexrelease>\let\filec@ntents@overwrite \@undefined
1365 <|latexrelease>\let\filec@ntents@noheader \@undefined
1366 <|latexrelease>\let\filec@ntents@nosearch \@undefined
1367 <|latexrelease>\let\filec@ntents@checkdir \@undefined
1368 <|latexrelease>\let\filec@ntents@where \@undefined
1369 <|latexrelease>
1370 <|latexrelease>\begingroup%
1371 <|latexrelease>\@tempcnta=1
1372 <|latexrelease>\loop
1373 <|latexrelease> \catcode\@tempcnta=12 %
1374 <|latexrelease> \advance\@tempcnta\@ne %
1375 <|latexrelease> \ifnum\@tempcnta<32 %
1376 <|latexrelease> \repeat
1377 <|latexrelease>\catcode`*=11 %
1378 <|latexrelease>\catcode`^~M\active%
1379 <|latexrelease>\catcode`^~L\active\let^~L\relax%
1380 <|latexrelease>\catcode`^~I\active%
1381 <|latexrelease>
1382 <|latexrelease>\gdef\filec@ntents#1{%
1383 <|latexrelease> \openin\@inputcheck#1 %
1384 <|latexrelease> \ifeof\@inputcheck%
1385 <|latexrelease> \@latex@warning{no@line}%
1386 <|latexrelease> {Writing file '\@currdir#1'}%
1387 <|latexrelease> \chardef\reserved@c15 %
1388 <|latexrelease> \ch@ck7\reserved@c\write%
1389 <|latexrelease> \immediate\openout\reserved@c#1\relax%
1390 <|latexrelease> \else%
1391 <|latexrelease> \closein\@inputcheck%

```

```

1392 <|latexrelease> \@latex@warning@no@line%
1393 <|latexrelease> {File '#1' already exists on the system.\MessageBreak%
1394 <|latexrelease> Not generating it from this source}%
1395 <|latexrelease> \let\write@gobbletwo%
1396 <|latexrelease> \let\closeout\gobble%
1397 <|latexrelease> \fi%
1398 <|latexrelease> \if@tempswa%
1399 <|latexrelease> \immediate\write\reserved@c{%
1400 <|latexrelease> \percentchar\percentchar\space%
1401 <|latexrelease> \expandafter@gobble/string\LaTeX2e file '#1'^^J%
1402 <|latexrelease> \percentchar\percentchar\space generated by the %
1403 <|latexrelease> '@currenvir' \expandafter@gobblefour/string\newenvironment^^J%
1404 <|latexrelease> \percentchar\percentchar\space from source '\jobname' on %
1405 <|latexrelease> \number\year/\two@digits\month/\two@digits\day.^^J%
1406 <|latexrelease> \percentchar\percentchar}%
1407 <|latexrelease> \fi%
1408 <|latexrelease> \let\do\makeother\dospecials%
1409 <|latexrelease> \count@ 128\relax%
1410 <|latexrelease> \loop%
1411 <|latexrelease> \catcode\count@ 11\relax%
1412 <|latexrelease> \advance\count@ \one%
1413 <|latexrelease> \ifnum\count@<\@ccclvi%
1414 <|latexrelease> \repeat%
1415 <|latexrelease> \edef\E{\backslashchar end\string{@currenvir\string}}%
1416 <|latexrelease> \edef\reserved@b{%
1417 <|latexrelease> \def\noexpand\reserved@b{%
1418 <|latexrelease> #####1\#####2\#####3\relax}%
1419 <|latexrelease> \reserved@b{%
1420 <|latexrelease> \ifx\relax##3\relax%
1421 <|latexrelease> \immediate\write\reserved@c{##1}%
1422 <|latexrelease> \else%
1423 <|latexrelease> \edef^^M{\noexpand\end{@currenvir}}%
1424 <|latexrelease> \ifx\relax##1\relax%
1425 <|latexrelease> \else%
1426 <|latexrelease> \@latex@warning{Writing text '##1' before %
1427 <|latexrelease> \string\end{@currenvir}\MessageBreak as last line of #1}%
1428 <|latexrelease> \immediate\write\reserved@c{##1}%
1429 <|latexrelease> \fi%
1430 <|latexrelease> \ifx\relax##2\relax%
1431 <|latexrelease> \else%
1432 <|latexrelease> \@latex@warning{%
1433 <|latexrelease> Ignoring text '##2' after \string\end{@currenvir}}%
1434 <|latexrelease> \fi%
1435 <|latexrelease> \fi%
1436 <|latexrelease> ^^M}%
1437 <|latexrelease> \catcode'\^L\active%
1438 <|latexrelease> \let\L@\undefined%
1439 <|latexrelease> \def^\L{\expandafter\ifx\csname L\endcsname\relax\fi ^^J^J}%
1440 <|latexrelease> \catcode'\^I\active%
1441 <|latexrelease> \let\I@\undefined%
1442 <|latexrelease> \def^\I{\expandafter\ifx\csname I\endcsname\relax\fi\space}%
1443 <|latexrelease> \catcode'\^M\active%
1444 <|latexrelease> \edef^^M##1^M{%

```

```
1446 \langle latexrelease\rangle \noexpand\reserved@b##1\relax}\}}%
1447 \langle latexrelease\rangle\endgroup%
1448 \langle latexrelease\rangle\EndIncludeInRelease
1449 (*2ekernel)

1450 \begingroup
1451 \catcode`|= \catcode`\%
1452 \catcode`\#=12
1453 \catcode`*=11
1454 \gdef\@percentchar{%}
1455 \gdef\endfilecontents{|
1456 \immediate\closeout\reserved@c
1457 \def\T##1##2##3{|
1458 \ifx##1\@undefined\else
1459 \@latex@warning@no@line{##2 has been converted to Blank ##3e}|
1460 \fi|
1461 \T\L{Form Feed}{Lin}|
1462 \T\I{Tab}{Spac}|
1463 \immediate\write\@unused{}|
1464 \global\let\endfilecontents*\endfilecontents
```

We no longer prevent the code to be used after begin document (no rollback needed for this change).

```
1465 \%{@onlypreamble\filecontents
1466 \%{@onlypreamble\endfilecontents
1467 \%{@onlypreamble\filecontents*
1468 \%{@onlypreamble\endfilecontents
1469 \endgroup
1470 \%{@onlypreamble\filecontents
(End of Section for filecontents)
```

(End definition for \filecontents and \endfilecontents.)

## 5 Package/class rollback mechanism

```
1471 </2ekernel>
1472 <*2ekernel | latexreleasefirst>
```

\pkgcls@debug For testing we have a few extra lines of code that by default do nothing but one can set \pkgcls@debug to \typeout to get extra info. Sometime in the future this will be dropped.

```
1473 <*tracerollback>
1474 %\let\pkgcls@debug\typeout
1475 \let\pkgcls@debug\gobble
1476 </tracerollback>
```

(End definition for \pkgcl{S@debug}.)

`\requestedLaTeXdate` The macro (!) `\requestedLaTeXdate` holds the globally requested rollback date (via `latexrelease`) or zero if no such request was made.

```
1477 \def\requestedLaTeXdate{\emptyset}
```

(End definition for \requestedLaTeXdate.)

\pkgcls@targetdate If a rollback for a package or class is requested then \pkgcls@targetdate holds the requested date as a number YYYYMMDD (if there was one, otherwise the value of \requestedLaTeXdate) and \pkgcls@targetlabel will be empty. If there was a request for a named version then \pkgcls@targetlabel holds the version name and \pkgcls@targetdate is set to 1.

\pkgcls@targetdate=0 is used to indicate that there was no rollback request. While loading an old release \pkgcls@targetdate is also reset to zero so that \DeclareRelease declarations are bypassed.

In contrast \pkgcls@innerdate will always hold the requested date (in a macro not a counter) if there was one, otherwise, e.g., if there was no request or a request to a version name it will contain  $\text{\TeX}$  largest legal number. While loading a file this can be used to provide conditionals that select code based on the request.

```
1478 \ifx\pkgcls@targetdate\@undefined
1479 \newcount\pkgcls@targetdate
1480 \fi
1481 \let\pkgcls@targetlabel\@empty
1482 \def\pkgcls@innerdate{\maxdimen}
```

(End definition for \pkgcls@targetdate, \pkgcls@targetlabel, and \pkgcls@innerdate.)

\pkgcls@candidate When looping through the \DeclareRelease declarations we record if the release is the best candidate we have seen so far. This is recorded in \pkgcls@candidate and we update it whenever we see a better one.

In \pkgcls@releasedate we keep track of the release date of that candidate.

```
1483 \let\pkgcls@candidate\@empty
1484 \let\pkgcls@releasedate\@empty
```

(End definition for \pkgcls@candidate and \pkgcls@releasedate.)

\load@onefilewithoptions the best place to add the rollback code is at the point where \onefilewithoptions is called to load a single class or package.

To make things easy we save the old definition as \load@onefilewithoptions and then provide a new interface.

Important: as this code is also unconditionally placed into latexrelease we can only do this name change once otherwise both macros will contain the same code.

```
1485 \ifx\load@onefilewithoptions\@undefined
1486 \let\load@onefilewithoptions\onefilewithoptions
1487 \def\@onefilewithoptions#1[#2] [#3] #4{%
```

First a bit of tracing normally disabled.

```
1488 (*tracerollback)
1489 \pkgcls@debug{--- File loaded request (\noexpand\usepackage or ...)}%
1490 \pkgcls@debug{\@spaces 1: #1}%
1491 \pkgcls@debug{\@spaces 2: #2}%
1492 \pkgcls@debug{\@spaces 3: #3}%
1493 \pkgcls@debug{\@spaces 4: #4}%
1494
```

Two of the arguments are needed later on in error/warning messages so we save them.

```
1495 \def\pkgcls@name{\#1}% % for info message
1496 \def\pkgcls@arg {\#3}% % for info message
```

then we parse the final optional argument to determine if there is a specific rollback request for the current file. This will set `\pkgcls@targetdate`, `\pkgcls@targetlabel` and `\pkgcls@mindate`.

```
1497 \pkgcls@parse@date@arg{#3}%
```

When determining the correct release to load we keep track of candidates in `\pkgcls@candidate` and initially we don't have any:

```
1498 \let\pkgcls@candidate\empty
```

If we had a rollback request then #3 may contain data but not necessarily a “minimal date” so instead of passing it on we pass on the content of `\pkgcls@mindate`. We need to pass the value not the command, otherwise nested packages may pick up the wrong information.

```
1499 \begingroup
1500 \edef\reserved@a{%
1501 \endgroup
1502 \unexpanded{\load@onefilewithoptions#1[#2]}%
1503 [\pkgcls@mindate]%
1504 \unexpanded{#4}%
1505 \reserved@a
1506 }
1507 \fi
```

(End definition for `\load@onefilewithoptions` and `\onefilewithoptions`.)

`\pkgcls@parse@date@arg` The `\pkgcls@parse@date@arg` command parses the second optional argument of `\usepackage`, `\RequirePackage` or `\documentclass` for a rollback request setting the values of `\pkgcls@targetdate` and `\pkgcls@targetlabel`.

This optional argument has a dual purpose: If it just contains a date string then this means that the package should have at least that date (to ensure that a certain feature is actually available, or a certain bug has been fixed). When the package gets loaded the information in `\Provides...` will then be checked against this request.

But if it starts with an equal sign followed by a date string or followed by a version name then this means that we should roll back to the state of the package at that date or to the version with the requested name.

If there was no optional argument or the optional argument does not start with “=” then the `\pkgcls@targetdate` is set to the date of the overall rollback request (via `latexrelease`) or if that was not given it is set to 0. In either case `\pkgcls@targetlabel` will be made empty.

If the argument doesn't start with “=” then it is supposed to be a “minimal date” and we therefore save the value in `\pkgcls@mindate`, otherwise this macro is made empty.

So in summary we have:

| Input                  | <code>\pkgcls@targetdate</code>             | <code>\pkgcls@targetlabel</code> | <code>\pkgcls@mindate</code> |
|------------------------|---------------------------------------------|----------------------------------|------------------------------|
| <code>\empty</code>    | <code>\global rollbackdate-as-number</code> | <code>\empty</code>              | <code>\empty</code>          |
| <code>\date</code>     | <code>\global rollbackdate-as-number</code> | <code>\empty</code>              | <code>\date</code>           |
| <code>=\date</code>    | <code>\date-as-number</code>                | <code>\empty</code>              | <code>\empty</code>          |
| <code>=\version</code> | 1                                           | <code>\version</code>            | <code>\empty</code>          |
| <code>\other</code>    | <code>\global rollbackdate-as-number</code> | <code>\empty</code>              | <code>\other</code>          |

where `\global rollbackdate-as-number` is a date request given via `latexrelease` or if there wasn't one 0.

```
1508 \def\pkgcls@parse@date@arg #1{%
```

If the argument is empty we use the rollback date from `\latexrelease` which has the value of zero if there was no rollback request. The label and the minimal date is made empty in that case.

```
1509 \ifx\@nil#1\@nil
1510 \pkgcls@targetdate\requestedLaTeXdate\relax
1511 \let\pkgcls@targetlabel\@empty
1512 \let\pkgcls@mindate\@empty
```

Otherwise we parse the argument further, checking for a = as the first character. We append a = at the end so that there is at least one such character in the argument.

```
1513 \else
1514 \pkgcls@parse@date@arg@#1=\@nil\relax
1515 \fi
1516 }
```

The actual parsing work then happens in `\pkgcls@parse@date@arg@:`

```
1517 \def\pkgcls@parse@date@arg@#1=#2\@nil{%
```

We set `\pkgcls@targetdate` depending on the parsing result; the code is expandable so we can do the parsing as part of the assignment.

```
1518 \pkgcls@targetdate
```

If a = was in first position then #1 will be empty. In that case #2 will be the original argument with a = appended.

This can be parsed with `\@parse@version`, the trailing character is simply ignored. This macro returns the parsed date as a number (or zero if it wasn't a date) and accepts both YYYY/MM/DD and YYYY-MM-DD formats.

```
1519 \ifx\@nil#1\@nil
1520 \@parse@version0#2//00\@nil\relax
```

Whatever is returned is thus assigned to `\pkgcls@targetdate` and therefore we can now test its value. If the value is zero we assume that the remaining argument string represents a version and change `\pkgcls@targetdate` and set `\pkgcls@targetlabel` to the version name (after stripping off the trailing =).

```
1521 \ifnum \pkgcls@targetdate=\z@
1522 \pkgcls@targetdate\@ne
1523 \def\pkgcls@innerdate{\maxdimen}%
1524 \pkgcls@parse@date@arg@version#2%
1525 \else
1526 \edef\pkgcls@innerdate{\the\pkgcls@targetdate}%
1527 \fi
1528 \let\pkgcls@mindate\@empty
1529 \else
```

If #1 was not empty then there wasn't a = character in first position so we are dealing either with a "minimum date" or with some incorrect data. We assume the former and make the following assignments (the first one finishing the assignment of `\pkgcls@targetdate`):

```
1530 \requestedLaTeXdate\relax
1531 \let\pkgcls@targetlabel\@empty
1532 \def\pkgcls@innerdate{\maxdimen}%
1533 \def\pkgcls@mindate{\#1}%
```

If the min-date is after the requested rollback date (if there is any, i.e., if it is not zero) then we have a conflict and therefore issue a warning.

```

1534 \ifnum \pkgcls@targetdate > \z@

1535 \ifnum \parse@version#1//00@nil > \pkgcls@targetdate

1536 @latex@warning@no@line{Suspicious rollback/min-date date given\MessageBreak

1537 A minimal date of #1 has been specified for

1538 \@cls@pkg\MessageBreak '\pkgcls@name'.\MessageBreak

1539 But this is in conflict

1540 with a rollback request to \requestedpatchdate}

1541 \fi

1542 \fi

1543 \fi

1544 }

Strip off the trailing = and assign the version name to \pkgcls@targetlabel.

1545 \def\pkgcls@parse@date@arg@version#1=%

1546 \def\pkgcls@targetlabel{-#1}

(End definition for \pkgcls@parse@date@arg.)
```

**\DeclareRelease** First argument is the “name” of the release and it can be left empty if one doesn’t like to give a name to the release. The second argument is that from which on this release was available (or should be used in case of minor updates). The final argument is the external file name of this release, by convention this should be *<pkg/cls-name>-<date>.⟨extension⟩* but this is not enforced and through this argument one can overwrite it.

```

1547 \def\DeclareRelease#1#2#3%

1548 \ifnum\pkgcls@targetdate>\z@ % some sort of rollback request

1549 (*tracerollback)

1550 \pkgcls@debug{---\string\DeclareRelease:}%"

1551 \pkgcls@debug{\@spaces 1: #1}"

1552 \pkgcls@debug{\@spaces 2: #2}"

1553 \pkgcls@debug{\@spaces 3: #3}"

1554 /*tracerollback)
```

If the date argument #2 is empty we are dealing with a special release that should be only accessible via its name; a typical use case would be a “beta” release. So if we are currently processing a date request we ignore it and otherwise we check if we can match the name and if so load the corresponding release file.

```

1555 \ifx\@nil#2\@nil

1556 \ifnum\pkgcls@targetdate=\@ne % named request

1557 \def\reserved@a{#1}"

1558 \ifx\pkgcls@targetlabel\reserved@a

1559 \pkgcls@use@this@release{#3}{}"

1560 (*tracerollback)

1561 \else

1562 \pkgcls@debug{Label doesn't match}"

1563 /*tracerollback)

1564 \fi

1565 (*tracerollback)

1566 \else

1567 \pkgcls@debug{Date request: ignored}"

1568 /*tracerollback)

1569 \fi

1570 \else
```

If the value of `\pkgcls@targetdate` is greater than 1 (or in reality greater than something like 19930101) we are dealing with a rollback request to a specific date.

```
1571 \ifnum\pkgcls@targetdate>\@ne % a real request
```

So we parse the date of this release to check if it is before or after the request date.

```
1572 \ifnum\@parse@version#2//00\@nil
1573 >\pkgcls@targetdate
```

If it is after we have to distinguish between two cases: If there was an earlier candidate we use that one because the other is too late, but if there wasn't one (i.e., if current release is the oldest that exists) we use it as the best choice. However in that case something is wrong (as there shouldn't be a rollback to a date where a package used doesn't yet exist). So we make a complained to the user.

```
1574 \ifx\pkgcls@candidate\@empty
1575 \pkgcls@rollbackdate@error{#2}%
1576 \pkgcls@use@this@release{#3}{#2}%
1577 \else
1578 \pkgcls@use@this@release\pkgcls@candidate
1579 \pkgcls@releasedate
1580 \fi
1581 \else
```

Otherwise, if the release date of this version is before the target rollback and we record it as a candidate. But we don't use it yet as there may be another release which is still before the target rollback.

```
1582 \def\pkgcls@candidate{#3}%
1583 \def\pkgcls@releasedate{#2}%
1584 (*tracer rollback)
1585 \pkgcls@debug{New candidate: #3}%
1586 (/tracer rollback)
1587 \fi
1588 \else
```

If we end up in this branch we have a named version request. So we check if `\pkgcls@targetlabel` matches the current name and if yes we use this release immediately, otherwise we do nothing as a later declaration may match it.

```
1589 \def\reserved@a{#1}%
1590 \ifx\pkgcls@targetlabel\reserved@a
1591 \pkgcls@use@this@release{#3}{#2}%
1592 (*tracer rollback)
1593 \else
1594 \pkgcls@debug{Label doesn't match}%
1595 (/tracer rollback)
1596 \fi
1597 \fi
1598 \fi
1599 \fi
1600 }
```

*(End definition for \DeclareRelease.)*

`\pkgcls@use@this@release` If a certain release has been selected (stored in the external file given in #1) we need to input it and afterwards stop reading the current file.

```
1601 \def\pkgcls@use@this@release#1#2{%
```

Before that we record the selection made inside the transcript.

```
1602 \pkgcls@show@selection{#1}{#2}%
```

We then set the `\pkgcls@targetdate` to zero so that any `\DeclareRelease` or `\DeclareCurrentRelease` in the file we now load are bypassed<sup>32</sup> and then we finally load the correct release.

After loading that file we need to stop reading the current file so we issue `\endinput`. Note that the `\relax` before that is essential to ensure that the `\endinput` is only happening after the file has been fully processed, otherwise it would act after the first line of the `\@@input`!

```
1603 \pkgcls@targetdate\z@
1604 \@@input #1\relax
1605 \endinput
1606 }
```

(End definition for `\pkgcls@use@this@release`.)

#### `\pkgcls@show@selection`

This command records what selection was made. As that is needed in two places (and it is rather lengthy) it was placed in a separate command. The first argument is the name of the external file that is being loaded and is only needed for debugging. The second argument is the date that corresponds to this file and it is used as part of the message.

```
1607 \def\pkgcls@show@selection#1#2{
1608 {*tracerollback}
1609 \pkgcls@debug{Result: use #1}%
1610 /{*tracerollback}
1611 \GenericInfo
1612 {\\@spaces\\@spaces\\space}{Rollback for
1613 \\@cls@pkg\\space'\\currname' requested ->
1614 \\ifnum\\pkgcls@targetdate>\\ne
1615 date
1616 \\ifnum\\requestedLaTeXdate=\\pkgcls@targetdate
1617 \\requestedpatchdate
1618 \\else
1619 \\expandafter\\gobble\\pkgcls@arg
1620 \\fi.\\MessageBreak
```

Instead of “best approximation” we could say that we have been able to exactly match the date (if it is exact), but that would mean extra tests without much gain, so not done.

```
1621 Best approximation is
1622 \\else
1623 version '\\pkgcls@targetlabel'.\\MessageBreak
1624 This corresponds to
1625 \\fi
1626 \\ifx\\nil#2\\nil
1627 a special release%
1628 \\else
1629 the release introduced on #2%
1630 \\fi
1631 \\gobble}%
1632 }
```

---

<sup>32</sup>The older release may also have such declarations inside if it was a simply copy of the `.sty` or `.cls` file current at that date. Removing these declarations would make the file load a tiny bit faster, but this way it works in any case.

*(End definition for \pkgcls@show@selection.)*

\pkgcls@rollbackdate@error This is called if the requested rollback date is earlier than the earliest known release of a package or class.

A similar error is given if global rollback date and min-date on a specific package conflict with each other, but that case is happens only once so it is inlined.

```
1633 \def\pkgcls@rollbackdate@error#1{%
1634 \@latex@error{Suspicious rollback date given}%
1635 {The '\@cls@pkg\space'\@currname' has no rollback data
1636 before #1 which\MessageBreak
1637 is after your requested rollback date --- so
1638 something may be wrong here.\MessageBreak
1639 Continue and we use the earliest known release.}}}
```

*(End definition for \pkgcls@rollbackdate@error.)*

\DeclareCurrentRelease This declares the date (and possible name) of the current version of a package or class.

```
1640 \def\DeclareCurrentRelease#1#2{%
```

First we test if \pkgcls@targetdate is greater than zero, otherwise this code is bypassed (as there is no rollback request).

```
1641 \ifnum\pkgcls@targetdate>\z@ % some sort of rollback request
1642 (*tracerollback)
1643 \pkgcls@debug{---DeclareCurrentRelease}%
1644 \pkgcls@debug{ 1: #1}%
1645 \pkgcls@debug{ 2: #2}%
1646 (/tracerollback)
```

If the value is greater than 1 we have to deal with a date request, so we parse #2 as a date and compare it with \pkgcls@targetdate.

```
1647 \ifnum\pkgcls@targetdate>\z@ % a date request
1648 \ifnum\@parse@version#2//00@nil
1649 >\pkgcls@targetdate
```

If it is greater that means the release date if this file is later than the requested rollback date. Again we have two cases: If there was a previous candidate release we use that one as the current release is too young, but if there wasn't we have to use this release nevertheless as there isn't any alternative.

However this case can only happen if there is a \DeclareCurrentRelease but no declared older releases (so basically the use of the declaration is a bit dubious).

```
1650 \ifx\pkgcls@candidate@\empty
1651 \pkgcls@rollbackdate@error{#2}%
1652 \else
1653 \pkgcls@use@this@release\pkgcls@candidate
1654 \pkgcls@releasedate
1655 \fi
```

Otherwise the current file is the right release, so we record that in the transcript and then carry on.

```
1656 \else
1657 \pkgcls@show@selection{current version}{#2}%
1658 \fi
1659 \else % a label request
```

Otherwise we have a rollback request to a named version so we check if that fits the current name and if not give an error as this was the last possible opportunity.

```

1660 \def\reserved@a{#1}%
1661 \ifx\pkgcls@targetlabel\reserved@a
1662 \pkgcls@show@selection{current version}{#2}%
1663 \else
1664 \@latex@error{Requested version '\pkgcls@targetlabel' for
1665 '@cls@pkg@space' '@currname' is unknown}@\ehc
1666 \fi
1667 \fi
1668 \fi
1669 }
```

*(End definition for \DeclareCurrentRelease.)*

- \IfTargetDateBefore This enables a simple form of conditional code inside a class or package file. If there is a date request and the request date is earlier than the first argument the code in the second argument is processed otherwise the code in the third argument is processed. If there was no date request then we also execute the third argument, i.e., we will get the “latest” version of the file.

Most often the second argument (before-date-code) will be empty.

```

1670 \DeclareRobustCommand\IfTargetDateBefore[1]{%
1671 \ifnum\pkgcls@innerdate <%
1672 \expandafter\@parse@version\expandafter#1//00@nil
1673 \typeout{Exclude code introduced on #1}%
1674 \expandafter\@firstoftwo
1675 \else
1676 \typeout{Include code introduced on #1}%
1677 \expandafter\@secondoftwo
1678 \fi
1679 }
```

*(End definition for \IfTargetDateBefore.)*

```
1680 </2ekernel | latexreleasefirst>
```

## 6 After Preamble

Finally we declare a package that allows all the commands declared above to be \onlypreamble to be used after \begin{document}.

```

1681 <@afterpreamble>
1682 \NeedsTeXFormat{LaTeX2e}
1683 \ProvidesPackage{pkgindoc}
1684 [2020-08-08 v1.3m Package Interface in Document (DPC)]
1685 \def\reserved@a#1\do@classoptionslist#2\do\filec@ntents#3\relax{%
1686 \gdef\@preamblecmds{#1#3}}
1687 \expandafter\reserved@a\@preamblecmds\relax
1688 </@afterpreamble>
```

# File U

## ltkeys.dtx

### 1 Creating and using keyval options

As with any key–value input, using key–value pairs as package or class options has two parts: creating the key options and setting (using) them. Options created in this way *may* be used after package loading as general key–value settings: this will depend on the nature of the underlying code.

---

```
\DeclareKeys \DeclareKeys [{family}] {declarations}
```

Creates a series of options from a comma-separated *declarations* list. Each entry in this list is a key–value pair, with the *key* having one or more *properties*. A small number of “basic” *properties* are described below. The full range of properties, provided by l3keys, can also be used for more powerful processing. See `interface3` for the full details.

The basic properties provided here are

- `.code` — execute arbitrary code
- `.if` — sets a TeX `\if...` switch
- `.store` — stores a value in a macro
- `.usage` — defines whether the option can be given only when loading (`load`), in the preamble (`preamble`) or has no limitation on scope (`general`)

The part of the *key* before the *property* is the *name*, with the *value* working with the *property* to define the behaviour of the option.

For example, with

```
\DeclareKeys[mypkg]
{
 draft.if = @mypkg@draft ,
 draft.usage = preamble ,
 name.store = \@mypkg@name ,
 name.usage = load ,
 second-name.store = \@mypkg@other@name
}
```

three options would be created. The option `draft` can be given anywhere in the preamble, and will set a switch called `\if@mypkg@draft`. The option `name` can only be given during package loading, and will save whatever value it is given in `\@mypkg@name`. Finally, the option `second-name` can be given anywhere, and will save its value in `\@mypkg@other@name`.

Keys created *before* the use of `\ProcessKeyOptions`/`\ProcessKeyPackageOptions` act as package options.

---

```
\DeclareUnknownKeyHandler \DeclareUnknownKeyHandler [<family>] {{code}}
```

The function `\DeclareUnknownKeyHandler` may be used to define the behavior when an undefined key is encountered. The `<code>` will receive the unknown key name as #1 and the value as #2. These can then be processed as appropriate, e.g. by forwarding to another package.

---

```
\ProcessKeyOptions \ProcessKeyOptions [<family>]
```

The `\ProcessKeyOptions` function is used to check the current option list against the keys defined for `<family>`. Global (class) options and local (package) options are checked when this function is called in a package.

---

```
\ProcessKeyPackageOptions \ProcessKeyPackageOptions [<family>]
```

This function works in a similar manner to `\ProcessKeyOptions`. When used in a package, `\ProcessKeyPackageOptions` will not examine any global (class) class options available. In contrast, `\ProcessKeyOptions` does parse class options (in common with the classical `\ProcessOptions` command).

---

```
\SetKeys \SetKeys [<family>] {{keyvals}}
```

Sets (applies) the explicit list of `<keyvals>` for the `<family>`: if the latter is not given, the value of `\@currname` used. This command may be used within a package to set options before or after using `\ProcessKeyOptions`/`\ProcessKeyPackageOptions`.

## 1.1 Implementation of `ltkeys`

```
1 (@@=keys)
2 (*2ekernel)
3 \ExplSyntaxOn
```

## 1.2 Key properties

```
.code
.if
4 \group_begin:
.store
5 \cs_set_protected:Npn __keys_tmp:nn #1#2
.usage
6 {
7 \quark_if_recursion_tail_stop:n {#1}
8 \cs_new_eq:cc
9 { \c_keys_props_root_str . #2 }
10 { \c_keys_props_root_str . #1 }
11 __keys_tmp:nn
12 }
13 __keys_tmp:nn
14 { code:n } { code }
15 { legacy_if_set:n } { if }
16 { tl_set:N } { store }
17 { usage:n } { usage }
18 { \q_recursion_tail } { }
19 \q_recursion_stop
20 \group_end:
```

(End definition for `.code` and others.)

### 1.3 Main mechanism

```

21 \cs_generate_variant:Nn \clist_put_right:Nn { Nv }
\l_keys_options_clist A single list is used for all options, into which they are collected before processing.
22 \clist_new:N \l_keys_options_clist
(End definition for \l_keys_options_clist.)

```

---

#### \l\_keys\_options\_loading\_bool

Used to indicate we are in the loading phase: controls the outcome of warnings.

```
23 \bool_new:N \l_keys_options_loading_bool
```

`\_keys_options:Nn`   The main function calls functions to collect up the global and local options into `\l__keys_options_clist` before calling the underlying functions to actually do the processing. So that a suitable message is produced if the option is unknown, the special `unknown` key is set if it does not already exist for the current family, and is cleaned up afterwards if required. To allow the L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> layer to know this mechanism is active, and to deal with the key family not matching the file name, we store the family in all cases.

```

24 \cs_new_protected:Npn _keys_options:Nn #1#2
25 { _keys_options_expand_module:Nn _keys_options:nN {#2} #1 }
26 \cs_new_protected:Npn _keys_options:nN #1#2
27 {
28 \cs_gset_nopar:cpx { opt@fam@\currname.\current } {#1}
29 \cs_set_protected:Npn _keys_option_end: { }
30 \clist_clear:N \l_keys_options_clist
31 _keys_options_global:Nn #2 {#1}
32 _keys_options_local:
33 \keys_if_exist:nnF {#1} { unknown }
34 {
35 \keys_define:nn {#1}
36 {
37 unknown .code:n =
38 {
39 \msg_error:nnxx { keys } { option-unknown }
40 { \l_keys_key_str } { \currname }
41 }
42 }
43 \cs_set_protected:Npn _keys_option_end:
44 { \keys_define:nn {#1} { unknown .undefine: } }
45 }
46 \bool_set_true:N \l_keys_options_loading_bool
47 \keys_set:nV {#1} \l_keys_options_clist
48 \bool_set_false:N \l_keys_options_loading_bool
49 \AtEndOfPackage { \cs_set_eq:NN \unprocessedoptions \scan_stop: }
50 _keys_option_end:
51 _keys_options_loaded:n {#1}
52 }
53 \msg_new:nnnn { keys } { option-unknown }
54 { Unknown-option~'#1'~for~package~#2. }
55 {
56 LaTeX-has-been-asked-to-set-an-option-called~'#1'~
57 but~the~#2~package~has~not~created~an~option~with~this~name.

```

```
58 }
```

(End definition for `\_keys_options:Nn`, `\_keys_options:nN`, and `\_keys_options_end:..`)

`\_keys_options_global:Nn`

Global (class) options are handled differently for L<sup>A</sup>T<sub>E</sub>X 2<sub>E</sub> packages and classes. Hence this function is essentially a check on the current file type. The initial test is needed as L<sup>A</sup>T<sub>E</sub>X 2<sub>E</sub> allows variables to be equal to `\scan_stop:`, which is usually forbidden in `expl3` code.

```
59 \cs_new_protected:Npn _keys_options_global:Nn #1#2
60 {
61 \cs_if_eq:NNF \@classoptionslist \scan_stop:
62 {
63 \cs_if_eq:NNTF \@curr ext \@csextension
64 {
65 _keys_options_class:n {#2}
66 {
67 \bool_if:NT #1
68 {
69 _keys_options_package:n {#2}
70 }
71 }
72 }
73 }
74 }
```

(End definition for `\_keys_options_global:Nn`.)

`\_keys_options_class:n`

For classes, each option (stripped of any content after `=`) is checked for existence as a key. If found, the option is added to the combined list for processing. On the other hand, unused options are stored up in `\@unusedoptionlist`. Before any of that, though, there is a simple check to see if there is an `unknown` key. If there is, then *everything* will match and the mapping can be skipped.

```
71 \cs_new_protected:Npn _keys_options_class:n #1
72 {
73 \cs_if_free:cF { opt@ \@currname . \@curr ext }
74 {
75 \keys_if_exist:nnTF {#1} { unknown }
76 {
77 \clist_put_right:Nv \l__keys_options_clist
78 { opt@ \@currname . \@curr ext }
79 }
80 }
81 \clist_map_inline:cn { opt@ \@currname . \@curr ext }
82 {
83 \keys_if_exist:neTF
84 {#1} { _keys_remove_equals:n {##1} }
85 { \clist_put_right:Nn \l__keys_options_clist {##1} }
86 { \clist_put_right:Nn \@unusedoptionlist {##1} }
87 }
88 }
89 }
90 }
```

(End definition for `\_keys_options_class:n`.)

`\_keys_options_package:n`

For global options when processing a package, the tasks are slightly different from those for a class. The check is the same, but here there is nothing to do if the option is not applicable. Each valid option also needs to be removed from `\@unusedoptionlist`.

```

91 \cs_new_protected:Npn __keys_options_package:n #1
92 {
93 \clist_map_inline:Nn \classoptionslist
94 {
95 \keys_if_exist:neT {#1} { __keys_remove_equals:n {##1} }
96 {
97 \clist_put_right:Nn \l__keys_options_clist {##1}
98 \clist_remove_all:Nn \unusedoptionlist {##1}
99 }
100 }
101 }

```

(End definition for `\__keys_options_package:n`.)

`\__keys_options_local:` If local options are found, they are added to the processing list. L<sup>A</sup>T<sub>E</sub>X 2<sub>E</sub> stores options for each file in a macro which may or may not exist, hence the need to use `\cs_if_exist:c`.

```

102 \cs_new_protected:Npn __keys_options_local:
103 {
104 \cs_if_eq:NNF \currext \clsextension
105 {
106 \cs_if_exist:cT { \opt@ \currname . \currext }
107 {
108 \clist_put_right:Nv \l__keys_options_clist
109 { \opt@ \currname . \currext }
110 }
111 }
112 }

```

(End definition for `\__keys_options_local:..`)

`\__keys_remove_equals:n` As the name suggests, this is a simple function to remove an equals sign from the input.  
`\__keys_remove_equals:w` This is all wrapped up in an `n` function so that there will always be a sign available.

```

113 \cs_new:Npn __keys_remove_equals:n #1
114 { __keys_remove_equals:w #1 = \s__keys_stop }
115 \cs_new:Npn __keys_remove_equals:w #1 = #2 \s__keys_stop { \exp_not:n {#1} }

```

(End definition for `\__keys_remove_equals:n` and `\__keys_remove_equals:w`.)

## 1.4 The document interfaces

```
116 \cs_generate_variant:Nn \keys_define:nn { nx }
```

`\__keys_options_expand_module:Nn` To deal with active characters inside the module argument whilst also expanding that argument, we use a combination of c- and f-type expansion. This works as the definitions for active UTF-8 bytes contain an `\ifincharname` test.

```

117 \cs_new_protected:Npn __keys_options_expand_module:Nn #1#2
118 {
119 \cs:w __keys_options_expand_module:nN \use:e { \cs_end: {#2} } #1
120 }
121 \cs_new_protected:Npn __keys_options_expand_module:nN #1#2
122 { #2 {#1} }

```

(End definition for `\__keys_options_expand_module:Nn` and `\__keys_options_expand_module:nN`.)

**\DeclareKeys** Defining key options is quite straight-forward: we have an intermediate function to allow for potential set-up steps.

```
123 \NewDocumentCommand \DeclareKeys { O { \currname } +m }
124 { __keys_options_expand_module:Nn \keys_define:nn {#1} {#2} }
```

(End definition for `\DeclareKeys`. This function is documented on page 841.)

**\DeclareUnknownKeyHandler**

```
125 \NewDocumentCommand \DeclareUnknownKeyHandler { O { \currname } +m }
126 {
127 \cs_set_protected:cpn { __keys_unknown_handler_ #1 :nn } ##1##2 {#2}
128 __keys_options_expand_module:Nn \keys_define:nx {#1}
129 {
130 unknown .code:n =
131 \exp_not:N \exp_args:NV
132 \exp_not:c { __keys_unknown_handler_ #1 :nn }
133 \exp_not:N \l_keys_key_str {####1}
134 }
135 }
```

(End definition for `\DeclareUnknownKeyHandler`. This function is documented on page 842.)

**\ProcessKeyOptions** We need to deal with the older interface from l3keys2e here: it had a mandatory argument.  
**\ProcessKeyPackageOptions** We can mop that up using a look-ahead, and then exploit that information to determine whether the package option handling is set up for the new approach for clash handling.

```
136 \NewDocumentCommand \ProcessKeyOptions { O { \currname } }
137 { __keys_options:Nn \c_true_bool {#1} }
138 \NewDocumentCommand \ProcessKeyPackageOptions { O { \currname } }
139 { __keys_options:Nn \c_false_bool {#1} }
140 \Ononlypreamble \ProcessKeyOptions
141 \Ononlypreamble \ProcessKeyPackageOptions
```

(End definition for `\ProcessKeyOptions` and `\ProcessKeyPackageOptions`. These functions are documented on page 842.)

## 1.5 Option usage scope

`\__keys_options_loaded:n` Indicates that the load-time options for a package have been processed: once this has happened, make them unavailable either with a warning or an error.

```
142 \cs_new_protected:Npn __keys_options_loaded:n #1
143 {
144 \prop_get:NnNT \l_keys_usage_load_prop {#1} \l__keys_tmpa_tl
145 {
146 \clist_map_inline:Nn \l__keys_tmpa_tl
147 {
148 \keys_define:nn {#1}
149 {
150 ##1 .code:n =
151 __keys_options_loaded:nn {#1} {##1}
152 }
153 }
154 }
155 }
156 \cs_new_protected:Npn __keys_options_loaded:nn #1#2
```

```

157 {
158 \bool_if:NTF \l__keys_options_loading_bool
159 {
160 \msg_warning:nxxx { keys } { load-option-ignored }
161 { \use:c { opt@fam@\currname.\@currext } } {#2}
162 }
163 { \msg_error:nnnn { keys } { load-only } {#1} {#2} }
164 }
165 \msg_new:nnn { keys } { load-option-ignored }
166 { Package~"#1"~has~already~been~loaded;~ignoring~load-time~option~"#2". }
167 \msg_new:nnnn { keys } { load-only }
168 { Key~"#2"~may~only~be~used~in~the~during~loading~of~package~"#1". }
169 {
170 LaTeX-was~asked~to~set~a~key~called~"#2",~but~this~is~only~allowed~
171 in~the~optional~argument~when~loading~package~"#1".
172 }

```

(End definition for `\_keys_options_loaded:n` and `\_keys_options_loaded:nn`.)

Disable all preamble options in one shot.

```

173 \tl_gput_left:Nn \@kernel@after@begindocument
174 {
175 \prop_map_inline:Nn \l_keys_usage_preamble_prop
176 {
177 \clist_map_inline:nn {#2}
178 {
179 \keys_define:nn {#1}
180 {
181 ##1 .code:n =
182 \msg_error:nnn { keys } { preamble-only } {##1}
183 }
184 }
185 }
186 }
187 \msg_new:nnnn { keys } { preamble-only }
188 { Key~"#1"~may~only~be~used~in~the~preamble. }
189 {
190 LaTeX-was~asked~to~set~a~key~called~"#1",~but~this~is~only~allowed~
191 before~\begin{document}.~You~will~need~to~set~the~key~earlier.
192 }

```

## 1.6 General key setting

**\SetKeys** A simple wrapper.

```

193 \NewDocumentCommand \SetKeys { O { \currname } +m }
194 { _keys_options_expand_module:Nn \keys_set:nn {#1} {#2} }

```

(End definition for `\SetKeys`. This function is documented on page 842.)

```

195 \ExplSyntaxOff
196
```

# File V

## ltfilehook.dtx

### Contents

## 1 Introduction

### 1.1 Provided hooks

The code offers a number of hooks into which packages (or the user) can add code to support different use cases. Many hooks are offered as pairs (i.e., the second hook is reversed). Also important to know is that these pairs are properly nested with respect to other pairs of hooks.

There are hooks that are executed for all files of a certain type (if they contain code), e.g., for all “include files” or all “packages”, and there are also hooks that are specific to a single file, e.g., do something after the package `foo.sty` has been loaded.

### 1.2 General hooks for file reading

There are four hooks that are called for each file that is read using document-level commands such as `\input`, `\include`, `\usepackage`, etc. They are not called for files read using internal low-level methods, such as `\@input` or `\openin`.

---

`file/before`  
`file/.../before`  
`file/.../after`  
`file/after`

---

These are:

**`file/before`, `file/<file-name>/before`** These hooks are executed in that order just before the file is loaded for reading. The code of the first hook is used with every file, while the second is executed only for the file with matching `<file-name>` allowing you to specify code that only applies to one file.

**`file/<file-name>/after`, `file/after`** These hooks are after the file with name `<file-name>` has been fully consumed. The order is swapped (the specific one comes first) so that the `before` and `after` hooks nest properly, which is important if any of them involve grouping (e.g., contain environments, for example). Furthermore both hooks are reversed hooks to support correct nesting of different packages adding code to both `/before` and `/after` hooks.

So the overall sequence of hook processing for any file read through the user interface commands of L<sup>A</sup>T<sub>E</sub>X is:

```
\UseHook{<file/before>}\n\UseHook{<file/<file name>/before>}\n <file contents>\n\UseHook{<file/<file name>/after>}\n\UseHook{<file/after>}
```

The file hooks only refer to the file by its name and extension, so the *<file name>* should be the file name as it is on the filesystem with extension (if any) and without paths. Different from `\input` and similar commands, the `.tex` extension is not assumed in hook *<file name>*, so `.tex` files must be specified with their extension to be recognized. Files within subfolders should also be addressed by their name and extension only.

Extensionless files also work, and should then be given without extension. Note however that `\TeX` prioritizes `.tex` files, so if two files `foo` and `foo.tex` exist in the search path, only the latter will be seen.

When a file is input, the *<file name>* is available in `\CurrentFile`, which is then used when accessing the `file/<file name>/before` and `file/<file name>/after`.

---

`\CurrentFile` The name of the file about to be read (or just finished) is available to the hooks through `\CurrentFile` (there is no `\Expl3` name for it for now). The file is always provided with its extension, i.e., how it appears on your hard drive, but without any specified path to it. For example, `\input{sample}` and `\input{app/sample.tex}` would both have `\CurrentFile` being `sample.tex`.

---

`\CurrentFilePath` The path to the current file (complement to `\CurrentFile`) is available in `\CurrentFilePath` if needed. The paths returned in `\CurrentFilePath` are only user paths, given through `\input@path` (or `\Expl3`'s equivalent `\l_file_search_path_seq`) or by directly typing in the path in the `\input` command or equivalent. Files located by `\kpsewhich` get the path added internally by the `\TeX` implementation, so at the macro level it looks as if the file were in the current folder, so the path in `\CurrentFilePath` is empty in these cases (package and class files, mostly).

---

`\CurrentFileUsed` `\CurrentFilePathUsed` In normal circumstances these are identical to `\CurrentFile` and `\CurrentFilePath`. They will differ when a file substitution has occurred for `\CurrentFile`. In that case, `\CurrentFileUsed` and `\CurrentFilePathUsed` will hold the actual file name and path loaded by `\TeX`, while `\CurrentFile` and `\CurrentFilePath` will hold the names that were *asked for*. Unless doing very specific work on the file being read, `\CurrentFile` and `\CurrentFilePath` should be enough.

### 1.3 Hooks for package and class files

Commands to load package and class files (e.g., `\usepackage`, `\RequirePackage`, `\LoadPackageWithOptions`, etc.) offer the hooks from section 1.2 when they are used to load a package or class file, e.g., `file/array.sty/after` would be called after the `array` package got loaded. But as packages and classes form as special group of files, there are some additional hooks available that only apply when a package or class is loaded.

---

|                                 |                                                                                                                                                              |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>package/before</code>     | These are:                                                                                                                                                   |
| <code>package/after</code>      |                                                                                                                                                              |
| <code>package/.../before</code> | <code>package/before, package/after</code> These hooks are called for each package being loaded.                                                             |
| <code>package/.../after</code>  |                                                                                                                                                              |
| <code>class/before</code>       | <code>package/⟨name⟩/before, package/⟨name⟩/after</code> These hooks are additionally called if the package name is <code>⟨name⟩</code> (without extension). |
| <code>class/after</code>        |                                                                                                                                                              |
| <code>class/.../before</code>   | <code>class/before, class/after</code> These hooks are called for each class being loaded.                                                                   |
| <code>class/.../after</code>    |                                                                                                                                                              |
|                                 | <code>class/⟨name⟩/before, class/⟨name⟩/after</code> These hooks are additionally called if the class name is <code>⟨name⟩</code> (without extension).       |

---

All `/after` hooks are implemented as reversed hooks.

The overall sequence of execution for `\usepackage` and friends is therefore:

```
\UseHook{⟨package/before⟩}
\UseHook{⟨package/⟨package name⟩/before⟩}
 \UseHook{⟨file/before⟩}
 \UseHook{⟨file/⟨package name⟩.sty/before⟩}
 ⟨package contents⟩
 \UseHook{⟨file/⟨package name⟩.sty/after⟩}
 \UseHook{⟨file/after⟩}

code from \AtEndOfPackage if used inside the package

\UseHook{⟨package/⟨package name⟩/after⟩}
\UseHook{⟨package/after⟩}
```

and similar for class file loading, except that `package/` is replaced by `class/` and `\AtEndOfPackage` by `\AtEndOfClass`.

If a package or class is not loaded (or it was loaded before the hooks were set) none of the hooks are executed!

All class or package hooks involving the name of the class or package are implemented as one-time hooks, whereas all other such hooks are normal hooks. This allows for the following use case

```
\AddToHook{⟨package/varioref/after⟩}
 { ... apply my customizations if the package gets
 loaded (or was loaded already) ... }
```

without the need to first test if the package is already loaded.

## 1.4 Hooks for `\include` files

To manage `\include` files, L<sup>A</sup>T<sub>E</sub>X issues a `\clearpage` before and after loading such a file. Depending on the use case one may want to execute code before or after these `\clearpages` especially for the one that is issued at the end.

Executing code before the final `\clearpage`, means that the code is processed while the last page of the included material is still under construction. Executing code after it means that all floats from inside the include file are placed (which might have added further pages) and the final page has finished.

Because of these different scenarios we offer hooks in three places.<sup>33</sup> None of the hooks are executed when an `\include` file is bypassed because of an `\includeonly` declaration. They are, however, all executed if L<sup>A</sup>T<sub>E</sub>X makes an attempt to load the `\include` file (even if it doesn't exist and all that happens is "No file `<filename>.tex`").

---

`include/before`  
`include/.../before`  
`include/end`  
`include/.../end`  
`include/after`  
`include/.../after`

These are:

**`include/before, include/<name>/before`** These hooks are executed (in that order) after the initial `\clearpage` and after `.aux` file is changed to use `<name>.aux`, but before the `<name>.tex` file is loaded. In other words they are executed at the very beginning of the first page of the `\include` file.

**`include/<name>/end, include/end`** These hooks are executed (in that order) after L<sup>A</sup>T<sub>E</sub>X has stopped reading from the `\include` file, but before it has issued a `\clearpage` to output any deferred floats.

**`include/<name>/after, include/after`** These hooks are executed (in that order) after L<sup>A</sup>T<sub>E</sub>X has issued the `\clearpage` but before it has switched back writing to the main `.aux` file. Thus technically we are still inside the `\include` and if the hooks generate any further typeset material including anything that writes to the `.aux` file, then it would be considered part of the included material and bypassed if it is not loaded because of some `\includeonly` statement.<sup>34</sup>

All `include` hooks involving the name of the included file are implemented as one-time hooks (whereas all other such hooks are normal hooks).

## 1.5 High-level interfaces for L<sup>A</sup>T<sub>E</sub>X

We do not provide any additional wrappers around the hooks (like `filehook` or `scrlfile` do) because we believe that for package writers the high-level commands from the hook management, e.g., `\AddToHook`, etc. are sufficient and in fact easier to work with, given that the hooks have consistent naming conventions.

---

<sup>33</sup>If you want to execute code before the first `\clearpage` there is no need to use a hook—you can write it directly in front of the `\include`.

<sup>34</sup>For that reason another `\clearpage` is executed after these hooks which normally does nothing, but starts a new page if further material got added this way.

## 1.6 Internal interfaces for L<sup>A</sup>T<sub>E</sub>X

---

```
\declare@file@substitution \declare@file@substitution {{file}} {{replacement-file}}
\undeclare@file@substitution \undeclare@file@substitution {{file}}
```

---

If *<file>* is requested for loading replace it with *<replacement-file>*. *\CurrentFile* remains pointing to *<file>* but *\CurrentFileUsed* will show the file actually loaded.

The main use case for this declaration is to provide a corrected version of a package that can't be changed (due to its license) but no longer functions because of L<sup>A</sup>T<sub>E</sub>X kernel changes, for example, or to provide a version that makes use of new kernel functionality while the original package remains available for use with older releases.

The *\undeclare@file@substitution* declaration undoes a substitution made earlier.

*Please do not misuse this functionality and replace a file with another unless if really needed and only if the new version is implementing the same functionality as the original one!*

---

```
\disable@package@load \disable@package@load {{package}} {{alternate-code}}
\reenable@package@load \reenable@package@load {{package}}
```

---

If *<package>* is requested do not load it but instead run *<alternate-code>* which could issue a warning, error or any other code.

The main use case is for classes that want to restrict the set of supported packages or contain code that make the use of some packages impossible. So rather than waiting until the document breaks they can set up informative messages why certain packages are not available.

The function is only implemented for packages not for arbitrary files.

## 1.7 A sample package for structuring the log output

As an application we provide the package `structuredlog` that adds lines to the `.log` when a file is opened and closed for reading keeping track of nesting level as well. For example, for the current document it adds the lines

```
= (LEVEL 1 START) t1lmr.fd
= (LEVEL 1 STOP) t1lmr.fd
= (LEVEL 1 START) supp-pdf.mkii
= (LEVEL 1 STOP) supp-pdf.mkii
= (LEVEL 1 START) nameref.sty
== (LEVEL 2 START) refcount.sty
== (LEVEL 2 STOP) refcount.sty
== (LEVEL 2 START) gettitlestring.sty
== (LEVEL 2 STOP) gettitlestring.sty
= (LEVEL 1 STOP) nameref.sty
= (LEVEL 1 START) ltfilehook-doc.out
= (LEVEL 1 STOP) ltfilehook-doc.out
= (LEVEL 1 START) ltfilehook-doc.out
= (LEVEL 1 STOP) ltfilehook-doc.out
= (LEVEL 1 START) ltfilehook-doc.hd
```

```

= (LEVEL 1 STOP) ltfilehook-doc.hd
= (LEVEL 1 START) ltfilehook.dtx
== (LEVEL 2 START) ot1lmr.fd
== (LEVEL 2 STOP) ot1lmr.fd
== (LEVEL 2 START) omllmm.fd
== (LEVEL 2 STOP) omllmm.fd
== (LEVEL 2 START) omslmsy.fd
== (LEVEL 2 STOP) omslmsy.fd
== (LEVEL 2 START) omxmlmex.fd
== (LEVEL 2 STOP) omxmlmex.fd
== (LEVEL 2 START) umsa.fd
== (LEVEL 2 STOP) umsa.fd
== (LEVEL 2 START) umsb.fd
== (LEVEL 2 STOP) umsb.fd
== (LEVEL 2 START) ts1lmr.fd
== (LEVEL 2 STOP) ts1lmr.fd
== (LEVEL 2 START) t1lmss.fd
== (LEVEL 2 STOP) t1lmss.fd
= (LEVEL 1 STOP) ltfilehook.dtx

```

Thus if you inspect an issue in the .log it is easy to figure out in which file it occurred, simply by searching back for LEVEL and if it is a STOP then remove 1 from the level value and search further for LEVEL with that value which should then be the START level of the file you are in.

## 2 The Implementation

```

1 {*2ekernel}
2 {@@=filehook}

```

### 2.1 Document and package-level commands

`\CurrentFile`  
`\CurrentFilePath`  
`\CurrentFileUsed`  
`\CurrentFilePathUsed`

User-level macros that hold the current file name and file path. These are used internally as well because the code takes care to protect against a possible redefinition of these macros in the loaded file (it's necessary anyway to make hooks work with nested `\input`). The versions `\...Used` hold the *actual* file name and path that is loaded by L<sup>A</sup>T<sub>E</sub>X, whereas the other two hold the name as requested. They will differ in case there's a file substitution.

```

3 {/2ekernel}
4 {*2ekernel | latexrelease}
5 {latexrelease}\IncludeInRelease{2020/10/01}%
6 {latexrelease} {\CurrentFile}{Hook management file}%
7 \ExplSyntaxOn
8 \tl_new:N \CurrentFile
9 \tl_new:N \CurrentFilePath
10 \tl_new:N \CurrentFileUsed
11 \tl_new:N \CurrentFilePathUsed
12 \ExplSyntaxOff
13 {/2ekernel | latexrelease}
14 {latexrelease}\EndIncludeInRelease

```

```

15 \langle latexrelease \rangle \IncludeInRelease{0000/00/00}%
16 \langle latexrelease \rangle \CurrentFile\{Hook management file}\%
17 \langle latexrelease \rangle
18 \langle latexrelease \rangle \let \CurrentFile \@undefined
19 \langle latexrelease \rangle \let \CurrentFilePath \@undefined
20 \langle latexrelease \rangle \let \CurrentFileUsed \@undefined
21 \langle latexrelease \rangle \let \CurrentFilePathUsed \@undefined
22 \langle latexrelease \rangle
23 \langle latexrelease \rangle \EndIncludeInRelease
24 \{*2ekernel}

```

(End definition for `\CurrentFile` and others. These functions are documented on page 849.)

## 2.2 `expl3` helpers

```

25 \{/2ekernel}
26 \{*2ekernel | latexrelease}
27 \langle latexrelease \rangle \IncludeInRelease{2020/10/01}%
28 \langle latexrelease \rangle \{__filehook_file_parse_full_name:nN\}{File helpers}%
29 \ExplSyntaxOn

```

`\_\_filehook_file_parse_full_name:nN`  
`\_\_filehook_full_name:nn`

A utility macro to trigger `expl3`'s file-parsing and lookup, and return a normalized representation of the file name. If the queried file doesn't exist, no normalization takes place. The output of `\_\_filehook_file_parse_full_name:nN` is passed on to the #2—a 3-argument macro that takes the `\path`, `\base`, and `\ext` parts of the file name.

```

30 \cs_new:Npn __filehook_file_parse_full_name:nN #1
31 {
32 \exp_args:Nf \file_parse_full_name_apply:nn
33 {
34 \exp_args:Nf __filehook_full_name:nn
35 { \file_full_name:n {#1} } {#1}
36 }
37 }
38 \cs_new:Npn __filehook_full_name:nn #1 #2
39 {
40 \tl_if_empty:nTF {#1}
41 { \tl_trim_spaces:n {#2} }
42 { \tl_trim_spaces:n {#1} }
43 }

```

(End definition for `\_\_filehook_file_parse_full_name:nN` and `\_\_filehook_full_name:nn`.)

`\_\_filehook_if_no_extension:nTF`  
`\_\_filehook_drop_extension:N`

Some actions depend on whether the file extension was explicitly given, and sometimes the extension has to be removed. The macros below use `\_\_filehook_file_parse_full_name:nN` to split up the file name and either check if `\ext` (#3) is empty, or discard it.

```

44 \cs_new:Npn __filehook_if_no_extension:nTF #1
45 {
46 \exp_args:Ne \tl_if_empty:nTF
47 { \file_parse_full_name_apply:nN {#1} \use_iii:nnn }
48 }
49 \cs_new_protected:Npn __filehook_drop_extension:N #1
50 {
51 \tl_gset:Nx #1

```

```

52 {
53 \exp_args:NV __filehook_file_parse_full_name:nN #1
54 __filehook_drop_extension_aux:nnn
55 }
56 }
57 \cs_new:Npn __filehook_drop_extension_aux:nnn #1 #2 #3
58 { \tl_if_empty:nF {#1} { #1 / } #2 }

```

(End definition for `\__filehook_if_no_extension:nTF` and `\__filehook_drop_extension:N`.)

```

\g__filehook_input_file_seq
\l__filehook_internal_tl
__filehook_file_push:
__filehook_file_pop:
__filehook_file_pop_assign:nnnn

```

Yet another stack, to keep track of `\CurrentFile` and `\CurrentFilePath` with nested `\inputs`. At the beginning of `\InputIfFileExists`, the current value of `\CurrentFilePath` and `\CurrentFile` is pushed to `\g__filehook_input_file_seq`, and at the end, it is popped and the value reassigned. Some other places don't use `\InputIfFileExists` directly (`\include`) or need `\CurrentFile` earlier (`\@onefilewithoptions`), so these are manually used elsewhere as well.

```

59 \tl_new:N \l__filehook_internal_tl
60 \seq_if_exist:NF \g__filehook_input_file_seq
61 { \seq_new:N \g__filehook_input_file_seq }
62 \cs_new_protected:Npn __filehook_file_push:
63 {
64 \seq_gpush:Nx \g__filehook_input_file_seq
65 {
66 { \CurrentFilePathUsed } { \CurrentFileUsed }
67 { \CurrentFilePath } { \CurrentFile }
68 }
69 }
70 \cs_new_protected:Npn __filehook_file_pop:
71 {
72 \seq_gpop:NNTF \g__filehook_input_file_seq \l__filehook_internal_tl
73 { \exp_after:wN __filehook_file_pop_assign:nnnn \l__filehook_internal_tl }
74 {
75 \msg_error:nnn { latex2e } { should-not-happen }
76 { Tried~to~pop~from~an~empty~file~name~stack. }
77 }
78 }
79 \cs_new_protected:Npn __filehook_file_pop_assign:nnnn #1 #2 #3 #4
80 {
81 \tl_set:Nn \CurrentFilePathUsed {#1}
82 \tl_set:Nn \CurrentFileUsed {#2}
83 \tl_set:Nn \CurrentFilePath {#3}
84 \tl_set:Nn \CurrentFile {#4}
85 }
86 \ExplSyntaxOff

```

(End definition for `\g__filehook_input_file_seq` and others.)

```

87 ⟨/2ekernel | latexrelease⟩
88 ⟨latexrelease⟩\EndIncludeInRelease

```

When rolling forward the following expl3 functions may not be defined. If we roll back the code does nothing.

```

89 ⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
90 ⟨latexrelease⟩ {\file_parse_full_name_apply:nN}{Roll forward help}%
91 ⟨latexrelease⟩

```

```

92 ⟨latexrelease⟩\ExplSyntaxOn
93 ⟨latexrelease⟩\cs_if_exist:NF\file_parse_full_name_apply:nN
94 ⟨latexrelease⟩{
95 ⟨latexrelease⟩\cs_new:Npn \file_parse_full_name_apply:nN #1
96 ⟨latexrelease⟩ {
97 ⟨latexrelease⟩ \exp_args:Ne __file_parse_full_name_auxi:nN
98 ⟨latexrelease⟩ { __kernel_file_name_sanitize:n {#1} }
99 ⟨latexrelease⟩ }
100 ⟨latexrelease⟩\cs_new:Npn __file_parse_full_name_auxi:nN #1
101 ⟨latexrelease⟩ {
102 ⟨latexrelease⟩ __file_parse_full_name_area:nw { } #1
103 ⟨latexrelease⟩ / \s__file_stop
104 ⟨latexrelease⟩ }
105 ⟨latexrelease⟩\cs_new:Npn __file_parse_full_name_area:nw #1 #2 / #3 \s__file_stop
106 ⟨latexrelease⟩ {
107 ⟨latexrelease⟩ \tl_if_empty:nTF {#3}
108 ⟨latexrelease⟩ { __file_parse_full_name_base:nw { } #2 . \s__file_stop {#1} }
109 ⟨latexrelease⟩ { __file_parse_full_name_area:nw { #1 / #2 }
110 ⟨latexrelease⟩ #3 \s__file_stop }
111 ⟨latexrelease⟩ }
112 ⟨latexrelease⟩\cs_new:Npn __file_parse_full_name_base:nw #1 #2 . #3 \s__file_stop
113 ⟨latexrelease⟩ {
114 ⟨latexrelease⟩ \tl_if_empty:nTF {#3}
115 ⟨latexrelease⟩ {
116 ⟨latexrelease⟩ \tl_if_empty:nTF {#1}
117 ⟨latexrelease⟩ {
118 ⟨latexrelease⟩ \tl_if_empty:nTF {#2}
119 ⟨latexrelease⟩ { __file_parse_full_name_tidy:nnnN { } { } }
120 ⟨latexrelease⟩ { __file_parse_full_name_tidy:nnnN { .#2 } { } }
121 ⟨latexrelease⟩ }
122 ⟨latexrelease⟩ { __file_parse_full_name_tidy:nnnN {#1} { .#2 } }
123 ⟨latexrelease⟩ }
124 ⟨latexrelease⟩ { __file_parse_full_name_base:nw { #1 . #2 }
125 ⟨latexrelease⟩ #3 \s__file_stop }
126 ⟨latexrelease⟩ }
127 ⟨latexrelease⟩\cs_new:Npn __file_parse_full_name_tidy:nnnN #1 #2 #3 #4
128 ⟨latexrelease⟩ {
129 ⟨latexrelease⟩ \exp_args:Ne #4
130 ⟨latexrelease⟩ {
131 ⟨latexrelease⟩ \str_if_eq:nnF {#3} { / } { \use_none:n }
132 ⟨latexrelease⟩ #3 \prg_do_nothing:
133 ⟨latexrelease⟩ }
134 ⟨latexrelease⟩ { \use_none:n #1 \prg_do_nothing: }
135 ⟨latexrelease⟩ {#2}
136 ⟨latexrelease⟩ }
137 ⟨latexrelease⟩}
138 ⟨latexrelease⟩\ExplSyntaxOff
139 ⟨latexrelease⟩
140 ⟨latexrelease⟩\EndIncludeInRelease
141 ⟨*2ekernel⟩
142 ⟨@@=⟩

```

## 2.3 Declaring the file-related hooks

These hooks have names with three-parts that start with `file/`, `include/`, `class/` or `package/` and end with `/before` or `/after` (or `/end` in the case of `include/`). They are all generic hooks so will be declared only if code is added to them; this declaration is done for you automatically and, indeed, they should not be declared explicitly.

Those named `.../after` and `include/.../end` are, when code is added, declared as reversed hooks.

## 2.4 Patching L<sup>A</sup>T<sub>E</sub>X's \InputIfFileExists command

Most of what we have to do is adding `\UseHook` into several L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub>  core commands, because of some circular dependencies in the kernel we do this only now and not in `ltfiles`.

```
\InputIfFileExists \InputIfFileExists loads any file if it is available so we have to add the hooks
@input@file@exists@with@hooks file/before and file/after in the right places. If the file doesn't exist no hooks
\unqu@tefilef@und should be executed.
```

```
143 </2ekernel>
144 <latexrelease>\IncludeInRelease{2020/10/01}%
145 <latexrelease> {\InputIfFileExists}{Hook management (files)}%
146 <*2ekernel | latexrelease>

147 \let\InputIfFileExists@\undefined
148 \DeclareRobustCommand \InputIfFileExists[2]{%
149 \IfFileExists{#1}%
150 {%
151 \@expl@@@filehook@file@push@@
152 \@filehook@set@CurrentFile
```

We pre-expand `\@filef@und` so that in case another file is loaded in the true branch of `\InputIfFileExists`, these don't change their value meanwhile. This isn't a worry with `\CurrentFile...` because they are kept in a stack.

```
153 \expandafter\@swaptwoargs\expandafter
154 {\expandafter\@input@file@exists@with@hooks
155 \expandafter{\@filef@und}}%
156 {#2}%
157 \@expl@@@filehook@file@pop@@
158 }%
159 }
160 \def\@input@file@exists@with@hooks#1{%
```

If the file exists then `\CurrentFile` holds its name. But we can't rely on that still being true after the file has been processed. Thus for using the name in the file hooks we need to preserve the name and then restore it for the `file/.../after` hook.

The hook always refers to the file requested by the user. The hook is *always* loaded for `\CurrentFile` which usually is the same as `\CurrentFileUsed`. In the case of a file replacement, the `\CurrentFileUsed` holds the actual file loaded. In any case the file names are normalized so that the hooks work on the real file name, rather than what the user typed in.

`\expl3`'s `\file_full_name:n` normalizes the file name (to factor out differences in the `.tex` extension), and then does a file lookup to take into account a possible path from `\l_file_search_path_seq` and `\input@path`. However only the file name and extension

are returned so that file hooks can refer to the file by their name only. The path to the file is returned in `\CurrentFilePath`.

```

161 \edef\reserved@a{%
162 \@expl@@@filehook@file@pop@assign@nnnn
163 {\CurrentFilePathUsed}%
164 {\CurrentFileUsed}%
165 {\CurrentFilePath}%
166 {\CurrentFile}}%
167 \expandafter\swaptwoargs\expandafter{\reserved@a}%

```

Before adding to the file list we need to make all (letter) characters catcode 11, because several packages use constructions like

```

\filename@parse{<filename>}
\ifx\filename@ext\clsextension
 ...
\fi

```

and that doesn't work if `\filename@ext` is `\detokenized`. Making `\clsextension` a string doesn't help much because some packages define their own `\<prefix>@someextension` with normal catcodes. This is not entirely correct because packages loaded (somehow) with catcode 12 alphabetic tokens (say, as the result of a `\string` or `\detokenize` command, or from a `\TeX` string like `\jobname`) will have these character tokens incorrectly turned into letter tokens. This however is rare, so we'll go for the all-letters approach (grepping the packages in `\TeX` Live didn't bring up any obvious candidate for breaking with this catcode change).

```

168 {\\edef\reserved@a{\unqu@tefilef@und#1\@nil}%
169 \@addtofilelist{\string\makeletter\reserved@a}%
170 \UseHook{file/before}}%

```

The current file name is available in `\CurrentFile` so we use that in the specific hook.

```

171 \UseHook{file/\CurrentFile/before}%
172 \@@input #1% <- trailing space comes from \@filef@und
173 }%

```

And here, `\CurrentFile` is restored (by `\@expl@@@filehook@file@pop@assign@nnnn`) so we can use it once more.

```

174 \UseHook{file/\CurrentFile/after}%
175 \UseHook{file/after}%
176 \def\unqu@tefilef@und"#1" \@nil{#1}

```

Now declare the non-generic file hooks used above:

```

177 \NewHook{file/before}
178 \NewReversedHook{file/after}
179 {latexrelease}\EndIncludeInRelease
180 {/2ekernel | latexrelease}

```

Now define `\InputIfFileExists` to input #1 if it seems to exist. Immediately prior to the input, #2 is executed. If the file #1 does not exist, execute '#3'.

```

181 {latexrelease}\IncludeInRelease{2019/10/01}%
182 {latexrelease} {\InputIfFileExists}{Hook management (files)}%
183 {latexrelease}%
184 {latexrelease}\DeclareRobustCommand \InputIfFileExists[2]{%
185 {latexrelease} \IfFileExists{#1}%
186 {latexrelease} {%

```

```

187 〈\latexrelease〉 \expandafter\@swaptwoargs\expandafter
188 〈\latexrelease〉 { \@filef@und } { \#2\@addtolist{#1}\@@input } } }
189 〈\latexrelease〉 \let\@input@file@exists@with@hooks\@undefined
190 〈\latexrelease〉 \let\unqu@tefilef@und\@undefined
191 〈\latexrelease〉 \EndIncludeInRelease
192 〈\latexrelease〉 \IncludeInRelease{0000/00/00}%
193 〈\latexrelease〉 { \InputIfFileExists } { Hook management (files) } %
194 〈\latexrelease〉 \long\def \InputIfFileExists#1#2{%
195 〈\latexrelease〉 \IfFileExists{#1}%
196 〈\latexrelease〉 { \#2\@addtolist{#1}\@@input \@filef@und } }

```

Also undo the internal command as some packages unfortunately test for their existence instead of using \IfFormatAtLeastTF.

```

197 〈\latexrelease〉 \expandafter\let\csname InputIfFileExists \endcsname\@undefined
198 〈\latexrelease〉 \let\@input@file@exists@with@hooks\@undefined
199 〈\latexrelease〉 \let\unqu@tefilef@und\@undefined
200 〈\latexrelease〉 \EndIncludeInRelease
201 〈*2ekernel〉

```

(End definition for \InputIfFileExists, \@input@file@exists@with@hooks, and \unqu@tefilef@und.)

## 2.5 Declaring a file substitution

```

202 〈@=filehook〉
203 〈/2ekernel〉
204 〈*2ekernel | latexrelease〉
205 〈\latexrelease〉 \IncludeInRelease{2020/10/01}%
206 〈\latexrelease〉 { __filehook_subst_add:nn } { Declaring file substitution } %
207 〈ExplSyntaxOn

__filehook_subst_add:nn
__filehook_subst_remove:n
__filehook_subst_file_normalize:Nn
__filehook_subst_empty_name_chk:NN

__filehook_subst_add:nn declares a file substitution by doing a (global) definition
of the form \def\@file-subst@{file}{replacement}. The file names are properly
sanitised, and normalized with the same treatment done for the file hooks. That is, a
file replacement is declared by using the file name (and extension, if any) only, and the
file path should not be given. If a file name is empty it is replaced by .tex (the empty
csname is used to check that).

208 \cs_new_protected:Npn __filehook_subst_add:nn #1 #2
209 {
210 \group_begin:
211 \cs_set:cpx { } { \exp_not:o { \cs:w\cs_end: } }
212 \int_set:Nn \tex_escapechar:D { -1 }
213 \cs_gset:cpx
214 {
215 @file-subst@
216 __filehook_subst_file_normalize:Nn \use_i_i_iii:n {#1}
217 }
218 { __filehook_subst_file_normalize:Nn __filehook_file_name_compose:nnn
219 {#2} }
220 \group_end:
221 }
222 \cs_new_protected:Npn __filehook_subst_remove:n #1
223 {
224 \group_begin:

```

```

225 \cs_set:cpx { } { \exp_not:o { \cs:w\cs_end: } }
226 \int_set:Nn \tex_escapechar:D { -1 }
227 \cs_undefine:c
228 {
229 @file-subst@
230 __filehook_subst_file_normalize:Nn \use_ii_iii:nnn {#1}
231 }
232 \group_end:
233 }
234 \cs_new:Npn __filehook_subst_file_normalize:Nn #1 #2
235 {
236 \exp_after:wN __filehook_subst_empty_name_chk:NN
237 \cs:w \exp_after:wN \cs_end:
238 \cs:w __filehook_file_parse_full_name:nN {#2} #1 \cs_end:
239 }
240 \cs_new:Npn __filehook_subst_empty_name_chk:NN #1 #2
241 { \if_meaning:w #1 #2 .tex \else: \token_to_str:N #2 \fi: }
(End definition for __filehook_subst_add:nn and others.)

```

\use\_ii\_iii:nnn A variant of \use\_... to discard the first of three arguments.

*Todo: this should move to expl3*

```

242 \cs_gset:Npn \use_ii_iii:nnn #1 #2 #3 {#2 #3}
(End definition for \use_ii_iii:nnn.)
243 \ExplSyntaxOff
244 </2ekernel | latexrelease>
245 <latexrelease>\EndIncludeInRelease
246 <*2ekernel>

```

\declare@file@substitution For two internals we provide L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> names so that we can use them elsewhere in the kernel (and so that they can be used in packages if really needed, e.g., `scrlfile`).

```

247 </2ekernel>
248 <*2ekernel | latexrelease>
249 <latexrelease>\IncludeInRelease{2020/10/01}%
250 <latexrelease> {\declare@file@substitution}{File substitution}%
251 \ExplSyntaxOn
252 \cs_new_eq:NN \declare@file@substitution __filehook_subst_add:nn
253 \cs_new_eq:NN \undeclare@file@substitution __filehook_subst_remove:n
254 \ExplSyntaxOff
255 </2ekernel | latexrelease>
256 <latexrelease>\EndIncludeInRelease

```

We are not fully rolling back the file substitutions in case a rollback encounters a package that contains them, but is itself not setup for rollback. So we just bypass them and hope for the best.

```

257 <latexrelease>\IncludeInRelease{0000/00/00}%
258 <latexrelease> {\declare@file@substitution}{File substitution}%
259 <latexrelease>
260 <latexrelease>\let \declare@file@substitution \gobbletwo
261 <latexrelease>\let \undeclare@file@substitution \gobbleone
262 <latexrelease>
263 <latexrelease>\EndIncludeInRelease
264 <*2ekernel>

```

(End definition for `\declare@file@substitution` and `\undeclare@file@substitution`. These functions are documented on page 852.)

```
265 <@@=〉
266 \ExplSyntaxOff
```

## 2.6 Selecting a file (`\set@curr@file`)

```
\set@curr@file
\set@curr@file@nosearch
 @curr@file
 @curr@file@reqd
```

Now we hook into `\set@curr@file` to resolve a possible file substitution, and add `\Expl@@@filehook@set@curr@file@@nNN` at the end, after `\curr@file` is set.

A file name is built using `\expandafter\string\csname<filename>\endcsname` to avoid expanding utf8 active characters. The `\csname` expands the normalization machinery and the routine to resolve a file substitution, returning a control sequence with the same name as the file.

It happens that when `<filename>` is empty, the generated control sequence is `\csname\endcsname`, and doing `\string` on that results in the file `csnameendcsname.tex`. To guard against that we `\ifx`-compare the generated control sequence with the empty csname. To do so, `\csname\endcsname` has to be defined, otherwise it would be equal to `\relax` and we would have false positives. Here we define `\csname\endcsname` to expand to itself to avoid it matching the definition of some other control sequence.

```
267 </2ekernel〉
268 <*2ekernel | latexrelease〉
269 <latexrelease>\IncludeInRelease{2022/06/01}%
270 <latexrelease> {\set@curr@file}{Setting current file name}%
271 \def\set@curr@file{%
272 \begingroup
273 \set@curr@file@aux
274 \edef\set@curr@file@nosearch{%
275 \begingroup
276 \let\noexpand\input@path\noexpand\empty
277 \csname seq_clear:N\endcsname
278 \expandafter\noexpand\csname l_file_search_path_seq\endcsname
279 \noexpand\set@curr@file@aux}
280 \def\set@curr@file@aux#1{%
281 \escapechar\m@ne
282 \let\protect\string
283 \edef~{\string~}%
284 \expandafter\def\csname\expandafter\endcsname
285 \expandafter{\csname\endcsname}}%
```

Two file names are set here: `@curr@file@reqd` which is the file requested by the user, and `@curr@file` which should be the same, except when we have a file substitution, in which case it holds the actual loaded file. `@curr@file` is resolved first, to check if a substitution happens. If it doesn't, `\Expl@@@filehook@if@file@replaced@@TF` short-cuts and just copies `@curr@file`, otherwise the full normalization procedure is executed.

At this stage the file name is parsed and normalized, but if the input doesn't have an extension, the default `.tex` is *not* added to `\curr@file` because for applications other than `\input` (graphics, for example) the default extension may not be `.tex`. First check if the input has an extension, then if the input had no extension, call `\Expl@@@filehook@drop@extension@@N`. In case of a file substitution, `\curr@file` will have an extension.

```
286 \Expl@@@filehook@if@no@extension@@nTF{#1}%
```

```

287 {\@tempsw@true}{\@tempsw@false}%
288 \o@kernel@make@file@csname\o@curr@file
289 \oexpl@@@filehook@resolve@file@subst@@w {#1}%
290 \oexpl@@@filehook@if@file@replaced@@TF
291 {\@kernel@make@file@csname\o@curr@file@reqd
292 \oexpl@@@filehook@normalize@file@name@@w{#1}%
293 \if@tempswa \oexpl@@@filehook@drop@extension@N\o@curr@file@reqd \fi}%
294 {\\if@tempswa \oexpl@@@filehook@drop@extension@N\o@curr@file \fi
295 \global\let\o@curr@file@reqd\o@curr@file}%
296 \oexpl@@@filehook@clear@replacement@flag@%
297 \endgroup}
298 (//ekernel | latexrelease)
299 (latexrelease)\EndIncludeInRelease

300 (latexrelease)\IncludeInRelease{2021/06/01}%
301 (latexrelease) {\set@curr@file}{Setting current file name}%
302 (latexrelease)\def\set@curr@file#1{%
303 (latexrelease) \begingroup
304 (latexrelease) \escapechar\m@ne
305 (latexrelease) \let\protect\string
306 (latexrelease) \edef~{\string-}%
307 (latexrelease) \expandafter\def\csname\expandafter\endcsname
308 (latexrelease) \expandafter{\csname\endcsname}%
309 (latexrelease) \oexpl@@@filehook@if@no@extension@nTF{#1}%
310 (latexrelease) {\@tempsw@true}{\@tempsw@false}%
311 (latexrelease) \o@kernel@make@file@csname\o@curr@file
312 (latexrelease) \oexpl@@@filehook@resolve@file@subst@@w {#1}%
313 (latexrelease) \oexpl@@@filehook@if@file@replaced@@TF
314 (latexrelease) {\@kernel@make@file@csname\o@curr@file@reqd
315 (latexrelease) \oexpl@@@filehook@normalize@file@name@@w{#1}%
316 (latexrelease) \if@tempswa \oexpl@@@filehook@drop@extension@N\o@curr@file@reqd \fi}%
317 (latexrelease) {\\if@tempswa \oexpl@@@filehook@drop@extension@N\o@curr@file \fi
318 (latexrelease) \global\let\o@curr@file@reqd\o@curr@file}%
319 (latexrelease) \oexpl@@@filehook@clear@replacement@flag@%
320 (latexrelease) \endgroup}
321 (latexrelease)\let\set@curr@file@nosearch@\undefined
322 (latexrelease)\EndIncludeInRelease

323 (latexrelease)\IncludeInRelease{2020/10/01}%
324 (latexrelease) {\set@curr@file}{Setting current file name}%
325 (latexrelease)\def\set@curr@file#1{%
326 (latexrelease) \begingroup
327 (latexrelease) \escapechar\m@ne
328 (latexrelease) \expandafter\def\csname\expandafter\endcsname
329 (latexrelease) \expandafter{\csname\endcsname}%
330 (latexrelease) \oexpl@@@filehook@if@no@extension@nTF{#1}%
331 (latexrelease) {\@tempsw@true}{\@tempsw@false}%
332 (latexrelease) \o@kernel@make@file@csname\o@curr@file
333 (latexrelease) \oexpl@@@filehook@resolve@file@subst@@w {#1}%
334 (latexrelease) \oexpl@@@filehook@if@file@replaced@@TF
335 (latexrelease) {\@kernel@make@file@csname\o@curr@file@reqd
336 (latexrelease) \oexpl@@@filehook@normalize@file@name@@w{#1}%
337 (latexrelease) \if@tempswa \oexpl@@@filehook@drop@extension@N\o@curr@file@reqd \fi}%
338 (latexrelease) {\\if@tempswa \oexpl@@@filehook@drop@extension@N\o@curr@file \fi
339 (latexrelease) \global\let\o@curr@file@reqd\o@curr@file}%

```

```

340 〈latexrelease〉 \expl@@@filehook@clear@replacement@flag@@
341 〈latexrelease〉 \endgroup
342 〈latexrelease〉\let\set@curr@file@nosearch@\undefined
343 〈latexrelease〉\EndIncludeInRelease
344 〈latexrelease〉\IncludeInRelease{2019/10/01}%
345 〈latexrelease〉 {\set@curr@file}{Setting current file name}%
346 〈latexrelease〉\def\set@curr@file#1{%
347 〈latexrelease〉 \begingroup
348 〈latexrelease〉 \escapechar\m@ne
349 〈latexrelease〉 \xdef\@curr@file{%
350 〈latexrelease〉 \expandafter\expandafter\expandafter\unquote@name
351 〈latexrelease〉 \expandafter\expandafter\expandafter{%
352 〈latexrelease〉 \expandafter\string
353 〈latexrelease〉 \csname\@firstofone#1\@empty\endcsname} }%
354 〈latexrelease〉 \endgroup
355 〈latexrelease〉}
356 〈latexrelease〉\let\set@curr@file@nosearch@\undefined
357 〈latexrelease〉\EndIncludeInRelease
358 〈latexrelease〉\IncludeInRelease{0000/00/00}%
359 〈latexrelease〉 {\set@curr@file}{Setting current file name}%
360 〈latexrelease〉\let\set@curr@file@\undefined
361 〈latexrelease〉\let\set@curr@file@nosearch@\undefined
362 〈latexrelease〉\EndIncludeInRelease
363 〈*2ekernel〉

```

(End definition for `\set@curr@file` and others.)

```
\@filehook@set@CurrentFile
\@kernel@make@file@csname
\@set@curr@file@aux
```

*To do: This should get internalized using `\expl@` names*

```

364 〈/2ekernel〉
365 〈*2ekernel | latexrelease〉
366 〈latexrelease〉\IncludeInRelease{2020/10/01}%
367 〈latexrelease〉 {\@kernel@make@file@csname}{Make file csname}%
368 \def\@kernel@make@file@csname#1#2#3{%
369 \xdef#1{\expandafter\@set@curr@file@aux
370 \csname\expandafter#2\@firstofone#3\@nil\endcsname} }

```

This auxiliary compares `\langle filename \rangle` with `\csname\endcsname` to check if the empty `.tex` file was requested.

```

371 \def\@set@curr@file@aux#1{%
372 \expandafter\ifx\csname\endcsname#1%
373 .tex\else\string#1\fi}
```

Then we call `\expl@@@filehook@set@curr@file@onNN` once for `\@curr@file` to set `\CurrentFile(Path)Used` and once for `\@curr@file@reqd` to set `\CurrentFile(Path)`. Here too the slower route is only used if a substitution happened, but here `\expl@@@filehook@if@file@replaced@TF` can't be used because the flag is reset at the `\endgroup` above, so we check if `\@curr@file` and `\@curr@file@reqd` differ. This macro is issued separate from `\set@curr@file` because it changes `\CurrentFile`, and side-effects would quickly get out of control.

```

374 \def\@filehook@set@CurrentFile{%
375 \expl@@@filehook@set@curr@file@onNN{\@curr@file}%
376 \CurrentFileUsed\CurrentFilePathUsed
377 \ifx\@curr@file@reqd\@curr@file
```

```

378 \let\CurrentFile\CurrentFileUsed
379 \let\CurrentFilePath\CurrentFilePathUsed
380 \else
381 \Expl@@@filehook@set@curr@file@onNN{\@curr@file@reqd}%
382 \CurrentFile\CurrentFilePath
383 \fi}
384 (/2ekernel | latexrelease)
385 (latexrelease)\EndIncludeInRelease
386 (*2ekernel)

(End definition for \@filehook@set@CurrentFile, \@kernel@make@file@csname, and
\@set@curr@file@aux.)

```

\@\_set\_curr\_file:nNN When inputting a file, `\set@curr@file` does a file lookup (in `\input@path` and `\l_file_search_path_seq`) and returns the actual file name (`\base` plus `\ext`) in `\CurrentFileUsed`, and in case there's a file substitution, the requested file in `\CurrentFile` (otherwise both are the same). Only the base and extension are returned, regardless of the input (both `path/to/file.tex` and `file.tex` end up as `file.tex` in `\CurrentFile`). The path is returned in `\CurrentFilePath`, in case it's needed.

```

387 (/2ekernel)
388 (*2ekernel | latexrelease)
389 (latexrelease)\IncludeInRelease{2020/10/01}%
390 (latexrelease) {\@_set_curr_file:nNN}{Set curr file}%
391 \ExplSyntaxOn
392 (@=filehook)
393 \cs_new_protected:Npn __filehook_set_curr_file:nNN #1
394 {
395 \exp_args:Nf __filehook_file_parse_full_name:nN {#1}
396 __filehook_set_curr_file_assign:nnnNN
397 }
398 \cs_new_protected:Npn __filehook_set_curr_file_assign:nnnNN #1 #2 #3 #4 #5
399 {
400 \str_set:Nn #5 {#1}
401 \str_set:Nn #4 {#2#3}
402 }
403 \ExplSyntaxOff
404 (/2ekernel | latexrelease)
405 (latexrelease)\EndIncludeInRelease
406 (*2ekernel)

(End definition for \@_set_curr_file:nNN and \@_set_curr_file_assign:nnnNN.)

```

## 2.7 Replacing a file and detecting loops

Start by sanitizing the file with `\__filehook_file_parse_full_name:nN` then do `\__filehook_file_subst_begin:nnn{\path}{\name}{\ext}`.

```

407 (/2ekernel)
408 (*2ekernel | latexrelease)
409 (latexrelease)\IncludeInRelease{2020/10/01}%
410 (latexrelease) {__filehook_resolve_file_subst:w}{Replace files detect loops}%
411 \ExplSyntaxOn
412 \cs_new:Npn __filehook_resolve_file_subst:w #1 @nil
413 { __filehook_file_parse_full_name:nN {#1} __filehook_file_subst_begin:nnn }
414 \cs_new:Npn __filehook_normalize_file_name:w #1 @nil

```

```

415 { __filehook_file_parse_full_name:nN {#1} __filehook_file_name_compose:nnn }
416 \cs_new:Npn __filehook_file_name_compose:nnn #1 #2 #3
417 { \tl_if_empty:nF {#1} { #1 / } #2#3 }

Since the file replacement is done expandably in a \csname, use a flag to remember if
a substitution happened. We use this in \set@curr@file to short-circuit some of it in
case no substitution happened (by far the most common case, so it's worth optimizing).
The flag raised during the file substitution algorithm must be explicitly cleared after the
__filehook_if_file_replaced:TF conditional is no longer needed, otherwise further
uses of __filehook_if_file_replaced:TF will wrongly return true.
```

```

flag __filehook_file_replaced
__filehook_if_file_replaced:TF
__filehook_clear_replacement_flag:
418 \flag_new:n { __filehook_file_replaced }
419 \cs_new:Npn __filehook_if_file_replaced:TF #1 #2
420 { \flag_if_raised:nTF { __filehook_file_replaced } {#1} {#2} }
421 \cs_new_protected:Npn __filehook_clear_replacement_flag:
422 { \flag_clear:n { __filehook_file_replaced } }
```

First off, start by checking if the current file ( $\langle name \rangle + \langle ext \rangle$ ) has a declared substitution. If not, then just put that as the name (including a possible  $\langle path \rangle$  in this case): this is the default case with no substitutions, so it's the first to be checked. The auxiliary  $\__filehook_file_subst_tortoise_hare:nn$  sees that there's no replacement for #2#3 and does nothing else.

```

423 \cs_new:Npn __filehook_file_subst_begin:nnn #1 #2 #3
424 {
425 __filehook_file_subst_tortoise_hare:nn { #2#3 } { #2#3 }
426 { __filehook_file_name_compose:nnn {#1} {#2} {#3} }
427 }
428 \ExplSyntaxOff
429 {/2ekernel | latexrelease}
430 {latexrelease}\EndIncludeInRelease
431 {*2ekernel}
```

### 2.7.1 The Tortoise and Hare algorithm

If there is a substitution ( $\langle true \rangle$  in the first  $\cs_if_exist:cTF$  below), then first check if there is no substitution down the line: this should be the second most common case, of one file replaced by another. In that case just leave the substitution there and the job is done. If any substitution happens, then the  $\flag{__filehook_file_replaced}$  is raised (conditionally, because checking if a flag is raised is much faster than raising it over and over again).

If, however there are more substitutions, then we need to check for a possible loop in the substitutions, which would otherwise put TeX in an infinite loop if just an exhaustive expansion was used.

To detect a loop, the *Tortoise and Hare* algorithm is used. The name of the algorithm is an analogy to Aesop's fable, in which the Hare outruns a Tortoise. The two pointers here are the csnames which contains each file replacement, both of which start at the position zero, which is the file requested. In the inner part of the macro below,  $\__filehook_file_subst_loop:cc$  is called with  $\@file-subst@\langle file \rangle$  and  $\@file-subst@\@file-subst@\langle file \rangle$ ; that is, the substitution of  $\langle file \rangle$  and the substitution of that substitution: the Tortoise walks one step while the Hare walks two.

Within  $\__filehook_file_subst_loop:NN$  the two substitutions are compared, and if they lead to the same file it means that there is a loop in the substitutions. If there's

no loop, `\__filehook_file_subst_tortoise_hare:nn` is called again with the Tortoise at position 1 and the hare at 2. Again, the substitutions are checked ahead of the Hare pointer to check that it won't run too far; in case there is no loop in the declarations, eventually one of the `\cs_if_exist:cTF` below will go `<false>` and the algorithm will end; otherwise it will run until the Hare reaches the same spot as the tortoise and a loop is detected.

```

432 </2ekernel>
433 <*2ekernel | latexrelease>
434 <latexrelease>\IncludeInRelease{2020/10/01}%
435 <latexrelease> {__filehook_file_subst_tortoise_hare:nn}{Tortoise and Hare}%
436 \ExplSyntaxOn
437 \cs_new:Npn __filehook_file_subst_tortoise_hare:nn #1 #2 #3
438 {
439 \cs_if_exist:cTF { @file-subst@ #2 }
440 {
441 \flag_if_raised:nF { __filehook_file_replaced }
442 { \flag_raise:n { __filehook_file_replaced } }
443 \cs_if_exist:cTF { @file-subst@ \use:c { @file-subst@ #2 } }
444 {
445 __filehook_file_subst_loop:cc
446 { @file-subst@ #1 }
447 { @file-subst@ \use:c { @file-subst@ #2 } }
448 }
449 { \use:c { @file-subst@ #2 } }
450 }
451 { #3 }
452 }

```

This is just an auxiliary to check if a loop was found, and continue the algorithm otherwise. If a loop is found, the `.tex` file is used as fallback and `\__filehook_file_subst_cycle_error:cN` is called to report the error.

```

453 \cs_new:Npn __filehook_file_subst_loop:NN #1 #2
454 {
455 \token_if_eq_meaning:NNTF #1 #2
456 {
457 .tex
458 __filehook_file_subst_cycle_error:cN { @file-subst@ #1 } #1
459 }
460 { __filehook_file_subst_tortoise_hare:nn {#1} {#2} {#2} }
461 }
462 \cs_generate_variant:Nn __filehook_file_subst_loop:NN { cc }

```

Showing this type of error expandably is tricky, as we have a very limited amount of characters to show and a potentially large list. As a work around, several errors are printed, each showing one step of the loop, until all the error messages combined show the loop.

```

463 \cs_new:Npn __filehook_file_subst_cycle_error:NN #1 #2
464 {
465 \msg_expandable_error:nnff { latex2e } { file-cycle }
466 {#1} { \use:c { @file-subst@ #1 } }
467 \token_if_eq_meaning:NNF #1 #2
468 { __filehook_file_subst_cycle_error:cN { @file-subst@ #1 } #2 }
469 }

```

```

470 \cs_generate_variant:Nn __filehook_file_subst_cycle_error:NN { c }
 And the error message:
471 \msg_new:nnn { latex2e } { file-cycle }
472 { File-loop!~#1-replaced~by~#2... }

(End definition for __filehook_resolve_file_subst:w and others.)

473 \ExplSyntaxOff
474 </2ekernel | latexrelease>
475 <latexrelease>\EndIncludeInRelease
476 <*2ekernel>
477 <@C=›

```

## 2.8 Preventing a package from loading

We support the use case of preventing a package from loading but not any other type of files (e.g., classes).

```

\disable@package@load
\reenable@package@load
\@disable@packageload@do

\disable@package@load defines \@pkg-disable@{package} to expand to some code #2 instead of loading the package.

478 </2ekernel>
479 <*2ekernel | latexrelease>
480 <latexrelease>\IncludeInRelease{2020/10/01}%
481 <latexrelease> {\@disable@package@load}{Disable packages}%
482 \def\disable@package@load#1#2{%
483 \global\@namedef{\@pkg-disable@#1.\@pkextension}{#2}%
484 \def\@disable@packageload@do#1#2{%
485 \@ifundefined{\@pkg-disable@#1}{#2}{%
486 {\@nameuse{\@pkg-disable@#1}}}
}

\reenable@package@load undefines \@pkg-disable@{package} to reallow loading a package.

487 \def\reenable@package@load#1{%
488 \global\expandafter\let
489 \csname \@pkg-disable@#1.\@pkextension \endcsname \undefined}
490 </2ekernel | latexrelease>
491 <latexrelease>\EndIncludeInRelease
492 <latexrelease>\IncludeInRelease{0000/00/00}%
493 <latexrelease> {\@disable@package@load}{Disable packages}%
494 <latexrelease>
495 <latexrelease>\let\disable@package@load \undefined
496 <latexrelease>\let\@disable@packageload@do\undefined
497 <latexrelease>\let\reenable@package@load \undefined
498 <latexrelease>\EndIncludeInRelease
499 <*2ekernel>

(End definition for \disable@package@load, \reenable@package@load, and
\@disable@packageload@do. These functions are documented on page 852.)

```

## 2.9 High-level interfaces for L<sup>A</sup>T<sub>E</sub>X

None so far and the general feeling for now is that the hooks are enough. Packages like filehook, etc., may use them to set up their interfaces (samples are given below) but for the now the kernel will not provide any.

## 2.10 Internal commands needed elsewhere

Here we set up a few horrible (but consistent) L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub>  names to allow for internal commands to be used outside this module (and in parts that still use L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub>  syntax). We have to unset the @@ since we want double “at” sign in place of double underscores.

```

500 <@@=
501 </2ekernel>
502 <*2ekernel | latexrelease>
503 <latexrelease>\IncludeInRelease{2020/10/01}%
504 <latexrelease> {\Expl@@@filehook@if@no@extension@nTF}{2e tmp interfaces}%
505 \ExplSyntaxOn

506 \cs_new_eq:NN \Expl@@@filehook@if@no@extension@nTF
507 __filehook_if_no_extension:nTF

508 \cs_new_eq:NN \Expl@@@filehook@set@curr@file@nNN
509 __filehook_set_curr_file:nNN

510 \cs_new_eq:NN \Expl@@@filehook@resolve@file@subst@w
511 __filehook_resolve_file_subst:w

512 \cs_new_eq:NN \Expl@@@filehook@normalize@file@name@w
513 __filehook_normalize_file_name:w

514 \cs_new_eq:NN \Expl@@@filehook@if@file@replaced@TF
515 __filehook_if_file_replaced:TF

516 \cs_new_eq:NN \Expl@@@filehook@clear@replacement@flag@%
517 __filehook_clear_replacement_flag:

518 \cs_new_eq:NN \Expl@@@filehook@drop@extension@N
519 __filehook_drop_extension:N

520 \cs_new_eq:NN \Expl@@@filehook@file@push@%
521 __filehook_file_push:

522 \cs_new_eq:NN \Expl@@@filehook@file@pop@%
523 __filehook_file_pop:

524 \cs_new_eq:NN \Expl@@@filehook@file@pop@assign@nnnn
525 __filehook_file_pop_assign:nnnn

526 \ExplSyntaxOff

```

This one specifically has to be undefined because it is left over in the input stream from \InputIfFileExists and executed when latexrelease is loaded. It cannot be \let to \undefined otherwise it would error as well, so it is \let to \relax to be silently ignored when loading \latexrelease.

```

527 </2ekernel | latexrelease>
528 <latexrelease>\EndIncludeInRelease
529 <latexrelease>
530 <latexrelease>\IncludeInRelease{0000/00/00}%
531 <latexrelease> {\Expl@@@filehook@if@no@extension@nTF}{2e tmp interfaces}%
532 <latexrelease>\let\Expl@@@filehook@file@pop@\relax
533 <latexrelease>\EndIncludeInRelease
534 <*2ekernel>

```

This ends the kernel code in this file.

```
535 </2ekernel>
```

### 3 A sample package for structuring the log output

```
536 {*structuredlog}
537 <@=filehook>
538 \ProvidesExplPackage
539 {structuredlog}{\ltfilehookdate}{\ltfilehookversion}
540 {Structuring the TeX transcript file}
\g_filehook_nesting_level_int
Stores the current package nesting level.
541 \int_new:N \g_filehook_nesting_level_int
Initialise the counter with the number of files in the \currnamestack (the number of items divided by 3) minus one, because this package is skipped when printing to the log.
542 \int_gset:Nn \g_filehook_nesting_level_int
543 { (\tl_count:N \currnamestack) / 3 - 1 }

(End definition for \g_filehook_nesting_level_int.)
```

\\_filehook\_log\_file\_record:n This macro is responsible for increasing and decreasing the file nesting level, as well as printing to the log. The argument is either STOPTART or STOP and the action it takes on the nesting integer depends on that.

```
544 \cs_new_protected:Npn _filehook_log_file_record:n #1
545 {
546 \str_if_eq:nnT {#1} {START} { \int_gincr:N \g_filehook_nesting_level_int }
547 \iow_term:x
548 {
549 \prg_replicate:nn { \g_filehook_nesting_level_int } { = } ~
550 (LEVEL ~ \int_use:N \g_filehook_nesting_level_int \c_space_tl #1) ~
551 \CurrentFileUsed
```

If there was a file replacement, show that as well:

```
552 \str_if_eq:NNF \CurrentFileUsed \CurrentFile
553 { ~ (\CurrentFile \c_space_tl requested) }
554 \iow_newline:
555 }
556 \str_if_eq:nnT {#1} {STOP} { \int_gdecr:N \g_filehook_nesting_level_int }
557 }
```

Now just hook the macro above in the generic `file/before...`

```
558 \AddToHook{file/before}{ _filehook_log_file_record:n { START } }
```

...and `file/after` hooks. We don't want to install the `file/after` hook immediately, because that would mean it is the first time executed when the package finishes. We therefore put the declaration inside `\AddToHookNext` so that it gets only installed when we have left this package.

```
559 \AddToHookNext{file/after}
560 { \AddToHook{file/after}{ _filehook_log_file_record:n { STOP } } }
```

(End definition for \\_filehook\_log\_file\_record:n.)

```
561 <@=/
562
```

## 4 Package emulations

### 4.1 Package **atveryend** emulation

With the new hook management and the hooks in `\enddocument` all of `atveryend` is taken care of. We can make an emulation only here after the substitution functionality is available:

```
563 <*2ekernel>
564 \declare@file@substitution{atveryend.sty}{atveryend-ltx.sty}
565 </2ekernel>
```

Here is the package file we point to:

```
566 <*atveryend-ltx>
567 \ProvidesPackage{atveryend-ltx}
568 [2020/08/19 v1.0a
569 Emulation of the original atveryend package^^Jwith kernel methods]
```

Here are new definitions for its interfaces now pointing to the hooks in `\enddocument`

```
570 \newcommand\AfterLastShipout {\AddToHook{\enddocument/afterlastpage}}
571 \newcommand\AtVeryEndDocument {\AddToHook{\enddocument/afteraux}}
```

Next one is a bit of a fake, but the result should normally be as expected. If not, one needs to add a rule to sort the code chunks in `enddocument/info`.

```
572 \newcommand\AtEndAfterFileList{\AddToHook{\enddocument/info}}
573 \newcommand\AtVeryVeryEnd {\AddToHook{\enddocument/end}}
```

`\BeforeClearDocument` This one is the only one we don't implement or rather don't have a dedicated hook in the code.

```
574 \ExplSyntaxOn
575 \newcommand\BeforeClearDocument[1]
576 { \AtEndDocument{#1}
577 \atveryend@DEPRECATED{BeforeClearDocument \tl_to_str:n{#1}}
578 }
579 \cs_new:Npn\atveryend@DEPRECATED #1
580 { \iow_term:x{=====DEPRECATED~USAGE~#1=====}}
581 \ExplSyntaxOff

(End definition for \BeforeClearDocument.)
```

```
582 </atveryend-ltx>
```

# File W

## ltshipout.dtx

### Contents

## 1 Introduction

The code provides an interface to the `\shipout` primitive of TeX which is called when a finished pages is finally “shipped out” to the target output file, e.g., the `.dvi` or `.pdf` file. A good portion of the code is based on ideas by Heiko Oberdiek implemented in his packages `atbegshi` and `atenddvi` even though the interfaces are somewhat different.<sup>35</sup>

### 1.1 Overloading the `\shipout` primitive

---

`\shipout` With this implementation TeX’s shipout primitive is no longer available for direct use. Instead `\shipout` is running some (complicated) code that picks up the box to be shipped out regardless of how that is done, i.e., as a constructed `\vbox` or `\hbox` or as a box register.

It then stores it in a named box register. This box can then be manipulated through a set of hooks after which it is shipped out for real.

Each shipout that actually happens (i.e., where the material is not discarded for one or the other reason) is recorded and the total number is available in a readonly variable and in a L<sup>A</sup>T<sub>E</sub>X counter.

---

`\RawShipout` This command implements a simplified shipout that bypasses the foreground and background hooks, e.g., only `shipout/firstpage` and `shipout/lastpage` are executed and the total shipout counters are incremented.

The command doesn’t use `\ShipoutBox` but its own private box register so that it can be used inside of shipout hooks to do some additional shipouts while already in the output routine with the current page being stored in `\ShipoutBox`. It does have access to `\ShipoutBox` if it is used in `shipout/before` (or `shipout/after`) and can use its content.

It is safe to use it in `shipout/before` or `shipout/after` but not necessarily in the other `shipout/...` hooks as they are intended for special processing.

---

<sup>35</sup>Heiko’s interfaces are emulated by the kernel code, if a document requests his packages, so older documents will continue to work.

---

`\ShipoutBox`  
`\l_shipout_box`

This box register is called `\ShipoutBox` (alternatively available via the L3 name `\l_`-`\shipout_box`).

This box is a “local” box and assignments to it should be done only locally. Global assignments (as done by some packages with older code where this is box is known as 255) may work but they are conceptually wrong and may result in errors under certain circumstances.

During the execution of `shipout/before` this box contains the accumulated material for the page, but not yet any material added by other shipout hooks. During execution of `shipout/after`, i.e., after the shipout has happened, the box also contains any background or foreground material.

Material from the hooks `shipout/firstpage` or `shipout/lastpage` is not included (but only used during the actual shipout) to facilitate reuse of the box data (e.g., `shipout/firstpage` material should never be added to a later page of the output).

---

`\l_shipout_box_ht_dim`  
`\l_shipout_box_dp_dim`  
`\l_shipout_box_wd_dim`  
`\l_shipout_box_ht_plus_dp_dim`

The shipout box dimensions are available in the L3 registers `\l_shipout_box_ht_dim`, etc. (there are no L<sup>A</sup>T<sub>E</sub>X 2<sub>E</sub> names).<sup>36</sup> These variables can be used inside the hook code for `shipout/before`, `shipout/foreground` and `shipout/background` if needed.

## 1.2 Provided hooks

---

`shipout/before`  
`shipout/after`  
`shipout/foreground`  
`shipout/background`  
`shipout/firstpage`  
`shipout/lastpage`

The code for `\shipout` offers a number of hooks into which packages (or the user) can add code to support different use cases. These are:

`shipout/before` This hook is executed after the finished page has been stored in `\ShipoutBox` / `\l_shipout_box`). It can be used to alter that box content or to discard it completely (see `\DiscardShipoutBox` below).

You can use `\RawShipout` inside this hook for special use cases. It can make use of `\ShipoutBox` (which doesn’t yet include the background and foreground material).

**Note:** It is not possible (or say advisable) to try and use this hook to typeset material with the intention to return it to main vertical list, it will go wrong and give unexpected results in many cases—for starters it will appear after the current page not before or it will vanish or the vertical spacing will be wrong!

`shipout/background` This hook adds a picture environment into the background of the page with the (0,0) coordinate in the top-left corner using a `\unitlength` of 1pt.

It should therefore only receive `\put` commands or other commands suitable in a `picture` environment and the vertical coordinate values would normally be negative.

---

<sup>36</sup>Might need changing, but HO’s version as strings is not really helpful I think).

Technically this is implemented by adding a zero-sized `\hbox` as the very first item into the `\ShipoutBox` containing that `picture` environment. Thus the rest of the box content will overprint what ever is typeset by that hook.

**shipout/foreground** This hook adds a picture environment into the foreground of the page with the  $(0,0)$  coordinate in the top-left corner using a `\unitlength` of `1pt`.

Technically this is implemented by adding a zero-sized `\hbox` as the very last item into the `\ShipoutBox` and raising it up so that it still has its  $(0,0)$  point in the top-left corner. But being placed after the main box content it will be typeset later and thus overprints it (i.e., is in the foreground).

**shipout/firstpage** The material from this hook is executed only once at the very beginning of the first output page that is shipped out (i.e., not discarded at the last minute). It should only contain `\special` or similar commands needed to direct post processors handling the `.dvi` or `.pdf` output.<sup>37</sup>

This hook is added to the very first page regardless of how it is shipped out (i.e., with `\shipout` or `\RawShipout`).

**shipout/lastpage** The corresponding hook to add `\specials` at the very end of the output file. It is only executed on the very last page of the output file — or rather on the page that `LATEX` believes is the last one. Again it is executed regardless of the `shipout` method.

It may not be possible for `LATEX` to correctly determine which page is the last one without several reruns. If this happens and the hook is non-empty then `LATEX` will add an extra page to place the material and also request a rerun to get the correct placement sorted out.

**shipout/after** This hook is executed after a shipout has happened. If the shipout box is discarded this hook is not looked at.

You can use `\RawShipout` inside this hook for special use cases and the main `\ShipoutBox` is still available at this point (but in contrast to `shipout/before` it now includes the background and foreground material).

**Note:** Just like `shipout/before` this hook is not meant to be used for adding typeset material back to the main vertical list—it might vanish or the vertical spacing will be wrong!

As mentioned above the hook `shipout/before` is executed first and can manipulate the prepared shipout box stored in `\ShipoutBox` or set things up for use in `\write` during the actual shipout. It is even run if there was a `\DiscardShipoutBox` request in the document.

The other hooks (except `shipout/after`) are added inside `hboxes` to the box being shipped out in the following order:

|                                                   |                        |
|---------------------------------------------------|------------------------|
| <code>shipout/firstpage</code>                    | only on the first page |
| <code>shipout/background</code>                   |                        |
| <code>&lt;boxed content of \ShipoutBox&gt;</code> |                        |
| <code>shipout/foreground</code>                   |                        |
| <code>shipout/lastpage</code>                     | only on the last page  |

---

<sup>37</sup>In `LATEX 2ε` that was already existing, but implemented using a box register with the name `\@begindvbox`.

If any of the hooks has no code then that particular no box is added at that point.

Once the (page) box has been shipped out the `shipout/after` hook is called (while you are still inside the output routine). It is not called if the shipout box was discarded.

In a document that doesn't produce pages, e.g., only makes \typeouts, none of the hooks are ever executed (as there is no `\shipout`) not even the `shipout/lastpage` hook.

If `\RawShipout` is used instead of `\shipout` then only the hooks `shipout/firstpage` and `shipout/lastpage` are executed (on the first or last page), all others are bypassed.

### 1.3 Legacy L<sup>A</sup>T<sub>E</sub>X commands

---

```
\AtBeginDvi \AtBeginDvi {<code>}
\AtEndDvi
```

`\AtBeginDvi` is the existing L<sup>A</sup>T<sub>E</sub>X 2<sub>E</sub> interface to fill the `shipout/firstpage` hook. This is not really a good name as it is not just supporting `.dvi` but also `.pdf` output or `.xdv`.

`\AtEndDvi` is the counterpart that was not available in the kernel but only through the package `atenddvi`. It fills the `shipout/lastpage` hook.

Neither interface can set a code label but uses the current default label.

As these two wrappers have been available for a long time we continue offering them (but not enhancing them, e.g., by providing support for code labels).

For new code we strongly suggest using the high-level hook management commands directly instead of “randomly-named” wrappers. This will lead to code that is easier to understand and to maintain and it also allows you to set code labels if needed.

For this reason we do not provide any other “new” wrapper commands for the above hooks in the kernel, but only keep the existing ones for backward compatibility.

### 1.4 Special commands for use inside the hooks

---

```
\DiscardShipoutBox \AddToHookNext {shipout/before} {...\DiscardShipoutBox...}
\shipout_discard:
```

The `\DiscardShipoutBox` declaration (L3 name `\shipout_discard:`) requests that on the next shipout the page box is thrown away instead of being shipped to the `.dvi` or `.pdf` file.

Typical applications wouldn't do this unconditionally, but have some processing logic that decides to use or not to use the page.

Note that if this declaration is used directly in the document it may depend on the placement to which page it applies, given that L<sup>A</sup>T<sub>E</sub>X output routine is called in an asynchronous manner! Thus normally one would use this only as part of the `shipout/before` code.

*Todo: Once we have a new mark mechanism available we can improve on that and make sure that the declaration applies to the page that contains it.*

In the `atbegshi` package there are a number of additional commands for use inside the `shipout/before` hook. They should normally not be needed any more as one can instead simply add code to the hooks `shipout/before`, `shipout/background` or `shipout/foreground`.<sup>38</sup> If `atbegshi` gets loaded then those commands become available as public functions with their original names as given below.

---

<sup>38</sup>If that assumption turns out to be wrong it would be trivial to change them to public functions (right now they are private).

## 1.5 Provided $\text{\LaTeX}$ callbacks

---

`pre_shipout_filter` Under  $\text{\LaTeX}$  the `pre_shipout_filter` Lua callback is provided which gets called immediately before the shipout primitive gets invoked. The signature is

```
function(<node> head)
 return true
end
```

The `head` is the list node corresponding to the box to be shipped out. The return value should always be `true`.

## 1.6 Information counters

---

`\ ReadonlyShipoutCounter` `\ifnum\ ReadonlyShipoutCounter=...`  
`\g_shipout_READONLY_int` `\int_use:N \g_shipout_READONLY_int % expl3 usage`

This integer holds the number of pages shipped out up to now (including the one to be shipped out when inside the output routine). More precisely, it is incremented only after it is clear that a page will be shipped out, i.e., after the `shipout/before` hook (because that might discard the page)! In contrast `shipout/after` sees the incremented value.

Just like with the `page` counter its value is only accurate within the output routine. In the body of the document it may be off by one as the output routine is called asynchronously!

Also important: it *must not* be set, only read. There are no provisions to prevent that restriction, but if you manipulate it, chaos will be the result. To emphasize this fact it is not provided as a  $\text{\LaTeX}$  counter but as a  $\text{\TeX}$  counter (i.e., a command), so `\Alpha{\ReadonlyShipoutCounter}` etc, would not work.

---

`totalpages` `\arabic{totalpages}`  
`\g_shipout_totalpages_int` `\int_use:N \g_shipout_totalpage_int % expl3 usage`

In contrast to `\ReadonlyShipoutCounter`, the `totalpages` counter is a  $\text{\LaTeX}$  counter and incremented for each shipout attempt including those pages that are discarded for one or the other reason. Again `shipout/before` sees the counter before it is incremented. In contrast `shipout/after` sees the incremented value.

Furthermore, while it is incremented for each page, its value is never used by  $\text{\LaTeX}$ . It can therefore be freely reset or changed by user code, for example, to additionally count a number of pages that are not build by  $\text{\LaTeX}$  but are added in a later part of the process, e.g., cover pages or picture pages made externally.

Important: as this is a page-related counter its value is only reliable inside the output routine!

---

`\PreviousTotalPages` `\thetotalpages\PreviousTotalPages`

Command that expands to the number of total pages from the previous run. If there was no previous run or if used in the preamble it expands to 0. Note that this is a command and not a counter, so in order to display the number in, say, Roman numerals you have to assign its value to a counter and then use `\Roman` on that counter.

## 1.7 Debugging shipout code

---

```
\DebugShipoutsOn
\DebugShipoutsOff
\shipout_debug_on:
\shipout_debug_off:
```

\DebugShipoutsOn

Turn the debugging of shipout code on or off. This displays changes made to the shipout data structures.

*Todo: This needs some rationalizing and may not stay this way.*

## 2 Emulating commands from other packages

The packages in this section are no longer necessary, but as they are used by other packages, they are emulated when they are explicitly loaded with `\usepackage` or `\RequirePackage`.

Please note that the emulation only happens if the package is explicitly requested, i.e., the commands documented below are not automatically available in the L<sup>A</sup>T<sub>E</sub>X kernel! If you write a new package we suggest to use the appropriate kernel hooks directly instead of loading the emulation.

### 2.1 Emulating `atbegshi`

---

```
\AtBeginShipoutUpperLeft \AddToHook {shipout/before}
\AtBeginShipoutUpperLeftForeground {\dots\AtBeginShipoutUpperLeft{\langle code\rangle}\dots}
```

This adds a `picture` environment into the background of the shipout box expecting `\langle code\rangle` to contain `picture` commands. The same effect can be obtained by simply using kernel features as follows:

```
\AddToHook{shipout/background}{\langle code\rangle}
```

There is one technical difference: if `\AtBeginShipoutUpperLeft` is used several times each invocation is put into its own box inside the shipout box whereas all `\langle code\rangle` going into `shipout/background` ends up all in the same box in the order it is added or sorted based on the rules for the hook chunks.

`\AtBeginShipoutUpperLeftForeground` is similar with the difference that the `picture` environment is placed in the foreground. To model it with the kernel functions use the hook `shipout/foreground` instead.

---

```
\AtBeginShipoutAddToBox \AddToHook {shipout/before} {\dots\AtBeginShipoutAddToBox{\langle code\rangle}\dots}
\AtBeginShipoutAddToBoxForeground
```

These work like `\AtBeginShipoutUpperLeft` and `\AtBeginShipoutUpperLeftForeground` with the difference that `\langle code\rangle` is directly placed into an `\hbox` inside the shipout box and not surrounded by a `picture` environment.

To emulate them using `shipout/background` or `shipout/foreground` you may have to wrap `\langle code\rangle` into a `\put` statement but if the code is not doing any typesetting just adding it to the hook should be sufficient.

---

`\AtBeginShipoutBox` This is the name of the shipout box as `atbegshi` knows it.

---

**\AtBeginShipoutOriginalShipout**

This is the name of the `\shipout` primitive as `atbegshi` knows it. This bypasses all the mechanisms set up by the L<sup>A</sup>T<sub>E</sub>X kernel and there are various scenarios in which it can therefore fail. It should only be used to run existing legacy `atbegshi` code but not in newly developed applications.

The kernel alternative is `\RawShipout` which is integrated with the L<sup>A</sup>T<sub>E</sub>X mechanisms and updates, for example, the `\ ReadonlyShipoutCounter` counter. Please use `\RawShipout` for new code if you want to bypass the before, foreground and background hooks.

---

**\AtBeginShipoutInit** By default `atbegshi` delayed its action until `\begin{document}`. This command was forcing it in an earlier place. With the new concept it does nothing.

---

**\AtBeginShipout** `\AtBeginShipout{\<code>}`  $\equiv$  `\AddToHook{shipout/before}{\<code>}`  
**\AtBeginShipoutNext** `\AtBeginShipoutNext{\<code>}`  $\equiv$  `\AddToHookNext{shipout/before}{\<code>}`

This is equivalent to filling the `shipout/before` hook by either using `\AddToHook` or `\AddToHookNext`, respectively.

---

**\AtBeginShipoutFirst** The `atbegshi` names for `\AtBeginDvi` and `\DiscardShipoutBox`.  
**\AtBeginShipoutDiscard**

## 2.2 Emulating `everyshi`

The `everyshi` package is providing commands to run arbitrary code just before the shipout starts. One point of difference: in the new shipout hooks the page is available as `\ShipoutBox` for inspection of change, one should not manipulate box 255 directly inside `shipout/before`, so old code doing this would change to use `\ShipoutBox` instead of 255 or `\@cclv`.

---

**\EveryShipout** `\EveryShipout{\<code>}`  $\equiv$  `\AddToHook{shipout/before}{\<code>}`

---

**\AtNextShipout** `\AtNextShipout{\<code>}`  $\equiv$  `\AddToHookNext{shipout/before}{\<code>}`

However, most use cases for `everyshi` are attempts to put some picture or text into the background or foreground of the page and that can be done today simply by using the `shipout/background` and `shipout/foreground` hooks without any need to coding.

## 2.3 Emulating `atenddvi`

The `atenddvi` package implemented only a single command: `\AtEndDvi` and that is now available out of the box so the emulation makes the package a no-op.

## 2.4 Emulating `everypage`

This package patched the original \begindvi hook and replaced it with its own version. Its functionality is now covered by the hooks offered by the kernel so that there is no need for such patching any longer.

\AddEverypageHook \AddEverypageHook{\{code\}} ≡  
    \AddToHook{shipout/background}{\put(1in,-1in){\{code\}}}  
  
\AddEverypageHook is adding something into the background of every page at a position of 1in to the right and 1in down from the top left corner of the page. By using the kernel hook directly you can put your material directly to the right place, i.e., use other coordinates in the \put statement above.

### 3 The Implementation

At the moment the whole module rolls back in one go, but if we make any modifications in later releases this will then need splitting.

```
2 <*2ekernel | latexrelease>
3 <latexrelease>\IncludeInRelease{2020/10/01}%
4 <latexrelease> \shipout{\Hook management (shipout)}%
5 \ExplSyntaxOn
```

### 3.1 Debugging

\g\_\_shipout\_debug\_bool Holds the current debugging state.

6 \bool\_new:N \g\_\_shipout\_debug\_bool

(End definition for `\g-_shipout_debug_bool`.)

```
\shipout_debug_on:
\shipout_debug_off:
 __shipout_debug:n
 __shipout_debug_gset:
 7 \cs_new_eq:NN __shipout_debug:n \use_none:n
 8 \cs_new_protected:Npn \shipout_debug_on:
 9 {
 10 \bool_gset_true:N \g__shipout_debug_bool
 11 __shipout_debug_gset:
 12 }
 13 \cs_new_protected:Npn \shipout_debug_off:
 14 {
 15 \bool_gset_false:N \g__shipout_debug_bool
 16 __shipout_debug_gset:
 17 }
 18 \cs_new_protected:Npn __shipout_debug_gset:
 19 {
 20 \cs_gset_protected:Npx __shipout_debug:n ##1
 21 { \bool_if:NT \g__shipout_debug_bool {##1} }
 22 }
```

(End definition for `\shipout_debug_on`: and others. These functions are documented on page 876.)

|                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\ShipoutBox</code>              | The box filled with the page to be shipped out (both L3 and L <sup>A</sup> T <sub>E</sub> X 2 <sub>&lt;</sub> name).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <code>\l_shipout_box</code>           | <pre>23 \box_new:N \l_shipout_box 24 \cs_set_eq:NN \ShipoutBox \l_shipout_box</pre> <p>(End definition for <code>\ShipoutBox</code> and <code>\l_shipout_box</code>. These functions are documented on page 872.)</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <code>\l__shipout_raw_box</code>      | The <code>\RawShipout</code> gets its own box but it is internal as there is no hook manipulation for it.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|                                       | <pre>25 \box_new:N \l__shipout_raw_box</pre> <p>(End definition for <code>\l__shipout_raw_box</code>.)</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>\__shipout_finalize_box:</code> | For LuaT <sub>E</sub> X invoke the <code>pre_shipout_filter</code> callback.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|                                       | <pre>26 \sys_if_engine_luatex:TF 27 { 28   \newprotectedluacmd \__shipout_finalize_box: 29   \exp_args:Nx \everyjob { 30     \exp_not:V \everyjob 31     \exp_not:N \lua_now:n { 32       luatexbase.create_callback('pre_shipout_filter', 'list') 33       local~call, getbox, setbox = luatexbase.call_callback, tex.getbox, tex.setbox~ 34       lua.get_functions_table()[\the\allocationnumber] = function() 35         local~head = getbox(\the\l_shipout_box) 36         local~result = call('pre_shipout_filter', head) 37         if-not (result == head) then~ 38           setbox(\the\l_shipout_box, result~or~nil) 39         end~ 40       end 41     } 42   } 43 } { 44   \cs_set_eq:NN \__shipout_finalize_box: \scan_stop: 45 }</pre> <p>(End definition for <code>\__shipout_finalize_box</code>.)</p> |
| <code>\__shipout_execute:</code>      | This is going to be the code run by <code>\shipout</code> . The code follows closely the ideas from atbegshi, so not documenting that here for now.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|                                       | <pre>46 \cs_set_protected:Npn \__shipout_execute: { 47   \tl_set:Nx \l__shipout_group_level_tl 48   { \int_value:w \tex_currentgrouplevel:D } 49   \tex_afterassignment:D \__shipout_execute_test_level: 50   \tex_setbox:D \l_shipout_box 51 }</pre> <p>(End definition for <code>\__shipout_execute</code>.)</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>\shipout</code>                 | Overloading the <code>\shipout</code> primitive:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|                                       | <pre>52 \cs_gset_eq:NN \shipout \__shipout_execute:</pre> <p>(End definition for <code>\shipout</code>. This function is documented on page 871.)</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

\l\_\_shipout\_group\_level\_t1 Helper token list to record the group level at which \\_\\_shipout\\_execute: is encountered.  
 53 \t1\_new:N \l\_\_shipout\_group\_level\_t1

(End definition for \l\_\_shipout\_group\_level\_t1.)

\\_\\_shipout\_execute\\_test\\_level: If the group level has changed then we are still constructing \l\\_shipout\\_box and to continue we need to wait until the current group has finished, hence the \tex\\_aftergroup:D.

```
54 \cs_new:Npn __shipout_execute_test_level: {
 55 \int_compare:nNnT
 56 \l__shipout_group_level_t1 < \tex_currentgrouplevel:D
 57 \tex_aftergroup:D __shipout_execute_cont:
 58 }
```

(End definition for \\_\\_shipout\_execute\_test\_level:.)

\\_\\_shipout\_execute\\_cont: This does the actual shipout running several hooks as part of it. The code for them is passed as argument #2 to #4 to \\_\\_shipout\\_execute\_main\\_cont:Nnnn; the first argument is the box to be shipped out.

```
59 \cs_new:Npn __shipout_execute_cont: {
 60 __shipout_execute_main_cont:Nnnn
 61 \l__shipout_box
 62 { \hook_use:n {shipout/before} }
 63 { \hook_if_empty:nF {shipout/foreground}
 64 { __shipout_add_foreground_picture:n
 65 { \hook_use:n {shipout/foreground} } } }
```

If the user hook for the background (shipout/background) has no code, there might still code in the kernel hook so we need to test for this too. We only test for the @kernel@before@shipout@background though. If the @kernel@after@shipout@background needs executing even if the user hook is empty then we can add another test (or the kernel could put something into the before hook).

```
66 \bool_lazy_and:nnF
 67 { \hook_if_empty_p:n {shipout/background} }
 68 { \t1_if_empty_p:N \@kernel@before@shipout@background }
 69 { __shipout_add_background_picture:n
 70 { \@kernel@before@shipout@background
 71 \hook_use:n {shipout/background}
 72 \@kernel@after@shipout@background } }
 73 }
 74 }
 75 { \hook_use:n {shipout/after} }
 76 }
```

(End definition for \\_\\_shipout\_execute\_cont:.)

\\_\\_shipout\_execute\_main\\_cont:Nnnn When we have reached this point the shipout box has been processed and is available in \l\\_shipout\\_box and ready for real ship out (unless it gets discarded during the process).

The three arguments hold hook code that is executed just before the actual shipout (#1), within the shipout adding background and foreground material (#2) and after the shipout has happened (#3). These are passed as arguments because the same code without those hooks is also used when doing a “raw” shipout implemented by \RawShipout. The only hook that is always executed is that for the very last page, i.e., shipout/lastpage.

First we quickly check if it is void (can't happen in the standard L<sup>A</sup>T<sub>E</sub>X output routine but `\shipout` might be called from a package that has some special processing logic). If it is void we aren't shipping anything out and processing ends.<sup>39</sup>

```
77 \cs_new:Npn __shipout_execute_main_cont:Nnnn #1#2#3#4 {
78 \box_if_empty:NTF #1
79 { \@latex@warning@no@line{ Ignoring~ void~ shipout~ box } }
80 { }
```

Otherwise we assume that we will ship something and prepare for final adjustments (in particular setting the state of `\protect` while we are running the hook code). We also save the current `\protect` state to restore it later.

```
81 % \bool_gset_false:N \g__shipout_discard_bool % setting this would disable
82 % % \DiscardShipoutBox on doc-level
83 \cs_set_eq:NN __shipout_saved_protect: \protect
84 \set@typeset@protect
```

We also store the current shipout box dimension in registers, so that they can be used in the hook code.<sup>40</sup>

```
85 __shipout_get_box_size:N #1
```

Then we execute the `shipout/before` hook (or nothing in case of `\RawShipout`).

```
86 #2
```

In `\g_shipout_totalpages_int` we count all shipout attempts so we increment that counter already here (the other one is incremented later when we know for sure that we do a `\shipout`).

We increment it after running the above hook so that the values for `\g_shipout_totalpages_int` and `\g_shipout_READONLY_int` are in sync while the hook is executed (in the case that totalpages isn't manually altered or through discarding pages that is).

```
87 \int_gincr:N \g_shipout_totalpages_int
```

The above hook might contain code that requests the page to be discarded so we now test for it.

```
88 \bool_if:NTF \g__shipout_discard_bool
89 { \@latex@info@no@line{Completed~ page~ discarded}
90 \bool_gset_false:N \g__shipout_discard_bool }
```

As we are discarding the page box and not shipping anything out, we need to do some house cleaning and reset T<sub>E</sub>X's deadcycles so that it doesn't complain about too many calls to the OR without any shipout.

```
91 \tex_deadcycles:D \c_zero_int
```

*Todo: In atbegshi the box was dropped but is that actually needed? Or the resetting of `\protect` to its kernel value?*

```
92 % \group_begin:
93 % \box_set_eq_drop:NN #1 #1
94 % \group_end:
95 % \cs_set_eq:NN \protect \exp_not:N
96 }
```

---

<sup>39</sup>In that case we don't reset the deadcycles, that would be up to the OR processing logic to do.

<sup>40</sup>This is not really necessary as the code could access them via `\box_ht:N`, etc., but it is perhaps convenient.

Even if there was no explicit request to discard the box it is possible that the code for the hook `shipout/before` has voided the box (by mistake or deliberately). We therefore test once more but this time make it a warning, because the best practice way is to use the request mechanism.

```

97 { \box_if_empty:NTF #1
98 { \@latex@warning@no@line { Ignoring~ void~ shipout~ box.
99 \MessageBreak The~ shipout~ box~ was~ voided~ by~ hook~ code }
100 }

```

Finally, if the box is still non-empty we are nearly ready to ship it out. First we increment the total page counter so that we can later test if we have reached the final page according to our available information.<sup>41</sup>

```

101 {
102 \int_gincr:N \g_shipout_READONLY_int
103 __shipout_debug:n
104 \typeout{Absolute~ page~ == \int_use:N \g_shipout_READONLY_int
105 \space (target:~ \@abspospage@last)}
106 }

```

Then we store the box sizes again (as they may have changed) and then look at the hooks `shipout/foreground` and `shipout/background`. If either or both are non-empty we add a `picture` environment to the box (in the foreground and/or in the background) and execute the hook code inside that environment.

```
107 __shipout_get_box_size:N #1
```

The first hook we run is the `shipout/firstpage` hook. This is only done once, then the `\__shipout_run_firstpage_hook:` command redefines itself to do nothing. If the hook contains `\specials` for integration at the top of the page they will be temporarily stored in a safe place and added later with `\__shipout_add_firstpage_specials:`.

```
108 __shipout_run_firstpage_hook:
```

Run the hooks for background and foreground or, if this is called by `\RawShipout`, copy the box `\l__shipout_raw_box` to `\l__shipout_box` so that firstpage and lastpage material gets added if necessary (that is always done to `\l__shipout_box`).

```
109 #3
```

We then run `\__shipout_add_firstpage_specials:` that adds the content of the hook `shipout/firstpage` to the start of the first page (if non-empty). It is then redefined to do nothing on later pages.

```
110 __shipout_add_firstpage_specials:
```

Then we check if we have to add the `shipout/lastpage` hook or the corresponding kernel hook because we have reached the last page. This test will be false for all but one (and hopefully the correct) page.

```

111 \int_compare:nNnT \@abspospage@last = \g_shipout_READONLY_int
112 { \bool_lazy_and:nnF
113 { \hook_if_empty_p:n {shipout/lastpage} }
114 { \tl_if_empty_p:N \@kernel@after@shipout@lastpage }
115 { __shipout_debug:n { \typeout{Executing~ lastpage~ hook~
116 on~ page~ \int_use:N \g_shipout_READONLY_int } }
117 __shipout_add_foreground_box:n
118 { \UseHook{shipout/lastpage}

```

---

<sup>41</sup>Doing that earlier would be wrong because we might end up with the last page counted but discarded and then we have no place to add the final objects into the output file.

```

119 \@kernel@after@shipout@lastpage }
120 }
121 \bool_gset_true:N \g__shipout_lastpage_handled_bool
122 }
123 __shipout_finalize_box:

```

Finally we run the actual TeX primitive for shipout. As that will expand delayed \write statements inside the page in which protected commands should not expand we first change \protect to the appropriate definition for that case.

```

124 \cs_set_eq:NN \protect \exp_not:N
125 \tex_shipout:D \box_use:N \l_shipout_box

```

The \l\_shipout\_box may contain the firstpage material if this was the very first shipout. That makes it unsuitable for reuse in another shipout, so as a safety measure the next command resets \l\_shipout\_box to its earlier state if that is necessary. On later pages this is then a no-op.

```

126 __shipout_drop_firstpage_specials:

```

The shipout/after hook (if in #4) needs to run with \protected commands again being executed, because that hook will “typeset” material added at the top of the next page.

```

127 \set@typeset@protect
128 #4
129 }
130 }

```

Restore the value of \protect in case \shipout is called outside of the output routine (where it is automatically restored because of the implicit group).

```

131 \cs_set_eq:NN \protect __shipout_saved_protect:
132 }
133 }

```

*(End definition for \\_\_shipout\_execute\_main\_cont:Nnnn.)*

\\_\_shipout\_execute\_raw:  
\\_\_shipout\_execute\_test\_level\_raw:

This implements the “raw” shipout which bypasses the before, foreground, background and after hooks. It follows the same pattern than \\_\_shipout\_execute\_raw: except that it finally calls \\_\_shipout\_execute\_main\_cont:Nnnn with three empty arguments. instead of the hook code.

```

134 \cs_set_protected:Npn __shipout_execute_raw: {
135 \tl_set:Nx \l__shipout_group_level_tl
136 { \int_value:w \tex_currentgrouplevel:D }
137 \tex_afterassignment:D __shipout_execute_test_level_raw:
138 \tex_setbox:D \l__shipout_raw_box
139 }
140 \cs_new:Npn __shipout_execute_test_level_raw: {
141 \int_compare:nNnT
142 \l__shipout_group_level_tl < \tex_currentgrouplevel:D
143 \tex_aftergroup:D __shipout_execute_nohooks_cont:
144 }

```

Well, not totally empty arguments, we add some debugging if we are actually doing a shipout.

```

145 \cs_new:Npn __shipout_execute_nohooks_cont: {
146 __shipout_execute_main_cont:Nnnn \l__shipout_raw_box
147 {} { __shipout_debug:n{ \typeout{Doing~ raw~ shipout~ ...} }
148 \box_set_eq:NN \l_shipout_box \l__shipout_raw_box } {}
149 }

```

(End definition for `\__shipout_execute_raw`: and `\__shipout_execute_test_level_raw`.)

`\RawShipout` The interface name for raw shipout.

150 `\cs_gset_eq:NN \RawShipout \__shipout_execute_raw:`

(End definition for `\RawShipout`. This function is documented on page 871.)

`\__shipout_saved_protect`: Remember the current `\protect` state.

151 `\cs_new_eq:NN \__shipout_saved_protect: \protect`

(End definition for `\__shipout_saved_protect`.)

`shipout/before` Declaring all hooks for the shipout code.

`shipout/after` 152 `\hook_new:n{shipout/before}`

`shipout/foreground` 153 `\hook_new:n{shipout/after}`

`shipout/background` 154 `\hook_new:n{shipout/foreground}`

`shipout/firstpage` 155 `\hook_new:n{shipout/background}`

`shipout/lastpage` 156 `\hook_new:n{shipout/firstpage}`

157 `\hook_new:n{shipout/lastpage}`

(End definition for `shipout/before` and others. These functions are documented on page 872.)

And here are the internal kernel hooks going before or after the public ones where needed.

158 `\let\@kernel@after@shipout@lastpage\@empty`

159 `\let\@kernel@before@shipout@background\@empty`

160 `\let\@kernel@after@shipout@background\@empty`

(End definition for `\@kernel@after@shipout@lastpage`, `\@kernel@before@shipout@background`, and `\@kernel@after@shipout@background`.)

`\__shipout_run_firstpage_hook`: There are three commands to handle the `shipout/firstpage` hook: `\__shipout_run_firstpage_hook`, `\__shipout_add_firstpage_specials`: and `\__shipout_drop_firstpage_specials`.

That hook is supposed to contain `\specials` and similar material to be placed at the very beginning of the output page and so it needs careful placing to avoid that anything else gets in front of it. And this means we have to wait with this until other hooks such as `shipout/background` have added their bits. It is also important that such `\specials` show up only on the very first page, so if this page gets saved before `\shipout` for later reuse, we have to make sure that they aren't in the saved version.

In addition the hook may also contain code to be executed "first", e.g., visible from code in `shipout/background` and this conflicts with adding the `\specials` late.

Therefore the processing is split into different parts: `\__shipout_run_firstpage_hook`: is done early and checks if there is any material in the hook.

161 `\cs_new:Npn \__shipout_run_firstpage_hook: {`

162 `\hook_if_empty:nTF {shipout/firstpage}`

If not then we define the other two commands to do nothing.

163 `{`

164 `\cs_gset_eq:NN \__shipout_add_firstpage_specials: \prg_do_nothing:`

165 `\cs_gset_eq:NN \__shipout_drop_firstpage_specials: \prg_do_nothing:`

166 `}`

If there is material we execute inside a box, which means any `\special` will end up in that box and any other code is executed and can have side effects (as long as they are global).

```
167 {
168 \hbox_set:Nn \l__shipout_firstpage_box { \UseHook{shipout/firstpage} }
169 }
```

Once we are here we change the definition to do nothing next time and we also change the command used to implement `\AtBeginDvi` to become a warning and not add further material to a hook that is never used again.

```
170 \cs_gset_eq:NN __shipout_run_firstpage_hook: \prg_do_nothing:
171 \cs_gset:Npn __shipout_add_firstpage_material:Nn ##1 ##2 {
172 @latex@warning{ First~ page~ is~ already~ shipped~ out,~ ignoring
173 \MessageBreak \string##1 }
174 }
175 }
```

*(End definition for `\__shipout_run_firstpage_hook`.)*

`\__shipout_add_firstpage_specials:` then adds the `\specials` stored in `\l__shipout_firstpage_box` to the page to be shipped out when the time is ready. Note that if there was no material in the `shipout/firstpage` hook then this command gets redefined to do nothing. But for most documents there is something, e.g., some PostScript header, or some meta data declaration, etc. so by default we assume there is something to do.

```
176 \cs_new:Npn __shipout_add_firstpage_specials: {
```

First we make a copy of the `\l_shipout_box` that we can restore it later on.

```
177 \box_set_eq:NN \l__shipout_raw_box \l_shipout_box
```

Adding something to the beginning means adding it to the background as that layer is done first in the output.

```
178 __shipout_add_background_box:n { \hbox_unpack_drop:N \l__shipout_firstpage_box }
```

After the actual shipout `\__shipout_drop_firstpage_specials:` is run to restore the earlier content of `\l_shipout_box` and then redefines itself again to do nothing.

As a final act we change the definition to do nothing next time.

```
179 \cs_gset_eq:NN __shipout_add_firstpage_specials: \prg_do_nothing:
180 }
```

The `\__shipout_drop_firstpage_specials:` is run after the shipout has occurred but before the `shipout/afterpage` hook is executed. That is the point where we have to restore the `\ShipoutBox` to its state without the `shipout/firstpage` material.

```
181 \cs_new:Npn __shipout_drop_firstpage_specials: {
182 \box_set_eq:NN \l_shipout_box \l__shipout_raw_box
```

If there was no such material then `\__shipout_run_firstpage_hook:` will have changed the definition to a no-op already. Otherwise this is what we do here.

```
183 \cs_gset_eq:NN __shipout_drop_firstpage_specials: \prg_do_nothing:
184 }
```

*(End definition for `\__shipout_add_firstpage_specials:` and  
`\__shipout_drop_firstpage_specials`.)*

`\l__shipout_firstpage_box` The box to hold any firstpage `\specials`.

```
185 \box_new:N \l__shipout_firstpage_box
```

(End definition for `\l__shipout_firstpage_box`.)

`\g__shipout_lastpage_handled_bool` A boolean to signal if we have already handled the `shipout/lastpage` hook.

186   `\bool_new:N \g__shipout_lastpage_handled_bool`

(End definition for `\g__shipout_lastpage_handled_bool`.)

`\_shipout_add_firstpage_material:Nn` This command adds material to the `shipout/firstpage` hook. It is used in `\AtBeginDvi`, etc. The first argument is the command through which it is called. Initially this is ignored but once we are passed the first page it can be used to generate a warning message mentioning the right user command.

187   `\cs_new:Npn \_shipout_add_firstpage_material:Nn #1#2 {`  
188       `\AddToHook{shipout/firstpage}{#2}`  
189   `}`

(End definition for `\_shipout_add_firstpage_material:Nn`.)

`\_shipout_get_box_size:N` Store the box dimensions in dimen registers.

*Todo:* This could/should perhaps be generalized to set height depth and width given an arbitrary box.

190   `\cs_new:Npn \_shipout_get_box_size:N #1 {`  
191       `\dim_set:Nn \l_shipout_box_ht_dim { \box_ht:N #1 }`  
192       `\dim_set:Nn \l_shipout_box_dp_dim { \box_dp:N #1 }`  
193       `\dim_set:Nn \l_shipout_box_wd_dim { \box_wd:N #1 }`  
194       `\dim_set:Nn \l_shipout_box_ht_plus_dp_dim`  
195       `{ \l_shipout_box_ht_dim + \l_shipout_box_dp_dim }`  
196   `}`

(End definition for `\_shipout_get_box_size:N`.)

`\l_shipout_box_ht_dim` And here are the variables set by `\_shipout_get_box_size:N`.

`\l_shipout_box_dp_dim`  
`\l_shipout_box_wd_dim`  
`\l_shipout_box_ht_plus_dp_dim`  
197   `\dim_new:N \l_shipout_box_ht_dim`  
198   `\dim_new:N \l_shipout_box_dp_dim`  
199   `\dim_new:N \l_shipout_box_wd_dim`  
200   `\dim_new:N \l_shipout_box_ht_plus_dp_dim`

(End definition for `\l_shipout_box_ht_dim` and others. These functions are documented on page 872.)

`\g__shipout_discard_bool` Indicate whether or not the current page box should be discarded

201   `\bool_new:N \g__shipout_discard_bool`

(End definition for `\g__shipout_discard_bool`.)

`\l__shipout_tmp_box` We need a box for the background and foreground material and a token register to remember badness settings as we disable them during the buildup below.

202   `\box_new:N \l__shipout_tmp_box`  
203   `\tl_new:N \l__shipout_saved_badness_tl`

(End definition for `\l__shipout_tmp_box` and `\l__shipout_saved_badness_tl`.)

`\_shipout_add_background_box:n` In standard L<sup>A</sup>T<sub>E</sub>X the shipout box is always a `\vbox` but here we are allow for other usage as well, in case some package has its own output routine.

204   `\cs_new:Npn \_shipout_add_background_box:n #1`  
205   `{ \_shipout_get_box_size:N \l_shipout_box`

But we start testing for a vertical box as that should be the normal case.

```
206 \box_if_vertical:NTF \l_shipout_box
207 {
```

Save current values of `\vfuzz` and `\vbadness` then change them to allow box manipulations without warnings.

```
208 \tl_set:Nx \l__shipout_saved_badness_tl
209 { \vfuzz=\the\vfuzz\relax
210 \vbadness=\the\vbadness\relax }
211 \vfuzz=\c_max_dim
212 \vbadness=\c_max_int
```

Then we reconstruct `\l_shipout_box` ...

```
213 \vbox_set_to_ht:Nnn \l_shipout_box \l_shipout_box_ht_plus_dp_dim
214 {
```

... the material in #1 is placed into a horizontal box with zero dimensions.

```
215 \hbox_set:Nn \l__shipout_tmp_box
216 { \l__shipout_saved_badness_tl #1 }
217 \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
218 \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
219 \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
```

The we typeset that box followed by whatever was in `\l_shipout_box` before (unpacked).

```
220 \skip_zero:N \baselineskip
221 \skip_zero:N \lineskip
222 \skip_zero:N \lineskiplimit
223 \box_use:N \l__shipout_tmp_box
224 \vbox_unpack:N \l_shipout_box
```

The `\kern` ensures that the box has no depth which is afterwards explicitly corrected.

```
225 \kern \c_zero_dim
226 }
227 \box_set_ht:Nn \l_shipout_box \l_shipout_box_ht_dim
228 \box_set_dp:Nn \l_shipout_box \l_shipout_box_dp_dim
```

*Todo: The whole boxing maneuver looks a bit like overkill to me, but for the moment I leave.*

```
229 \l__shipout_saved_badness_tl
230 }
231 {
```

A horizontal box is handled in a similar way. The last case would be a void box in which case we do nothing hence the missing F branch.

```
232 \box_if_horizontal:NT \l_shipout_box
233 {
234 \tl_set:Nx \l__shipout_saved_badness_tl
235 { \hfuzz=\the\hfuzz\relax
236 \hbadness=\the\hbadness\relax }
237 \hfuzz=\c_max_dim
238 \hbadness=\c_max_int
239 \hbox_set_to_wd:Nnn \l_shipout_box \l_shipout_box_wd_dim
240 {
241 \hbox_set:Nn \l__shipout_tmp_box
242 { \l__shipout_saved_badness_tl #1 }}
```

```

243 \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
244 \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
245 \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
246 \box_move_up:nn
247 \l_shipout_box_ht_dim
248 { \box_use:N \l__shipout_tmp_box }
249 \hbox_unpack:N \l_shipout_box
250 }
251 \l__shipout_saved_badness_tl
252 }
253 }
254 }
```

(End definition for `\_shipout_add_background_box:n`.)

`\_shipout_add_foreground_box:n` Foreground boxes are done in the same way, only the order and placement of boxes has to be done differently.

```

255 \cs_new:Npn _shipout_add_foreground_box:n #1
256 {
257 \box_if_vertical:NTF \l_shipout_box
258 {
259 \tl_set:Nx \l__shipout_saved_badness_tl
260 { \vfuzz=\the\vfuzz\relax
261 \vbadness=\the\vbadness\relax }
262 \vfuzz=\c_max_dim
263 \vbadness=\c_max_int
264 \vbox_set_to_ht:Nnn \l_shipout_box \l_shipout_box_ht_plus_dp_dim
265 {
266 \hbox_set:Nn \l__shipout_tmp_box
267 { \l__shipout_saved_badness_tl #1 }
268 \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
269 \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
270 \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
271 \skip_zero:N \baselineskip
272 \skip_zero:N \lineskip
273 \skip_zero:N \lineskiplimit
274 \vbox_unpack:N \l_shipout_box
275 \kern -\l_shipout_box_ht_plus_dp_dim
276 \box_use:N \l__shipout_tmp_box
277 \kern \l_shipout_box_ht_plus_dp_dim
278 }
279 \l__shipout_saved_badness_tl
280 \box_set_ht:Nn \l_shipout_box \l_shipout_box_ht_dim
281 \box_set_dp:Nn \l_shipout_box \l_shipout_box_dp_dim
282 }
283 {
284 \box_if_horizontal:NT \l_shipout_box
285 {
286 \tl_set:Nx \l__shipout_saved_badness_tl
287 { \hfuzz=\the\hfuzz\relax
288 \hbadness=\the\hbadness\relax }
289 \hfuzz=\c_max_dim
290 \hbadness=\c_max_int
291 \hbox_set_to_wd:Nnn \l_shipout_box \l_shipout_box_wd_dim
292 }
293 }
```

```

292 {
293 \hbox_unpack:N \l_shipout_box
294 \kern -\box_wd:N \l_shipout_box
295 \hbox_set:Nn \l_shipout_tmp_box
296 { \l_shipout_saved_badness_t1 #1 }
297 \box_set_wd:Nn \l_shipout_tmp_box \c_zero_dim
298 \box_set_ht:Nn \l_shipout_tmp_box \c_zero_dim
299 \box_set_dp:Nn \l_shipout_tmp_box \c_zero_dim
300 \box_move_up:nn { \box_ht:N \l_shipout_box }
301 { \box_use:N \l_shipout_tmp_box }
302 \kern \box_wd:N \l_shipout_box
303 }%
304 \l_shipout_saved_badness_t1
305 }
306 }
307 }
```

(End definition for `\_shipout_add_foreground_box:n`)

`\_shipout_init_page_origins:`

```
\c__shipout_horigin_tl
\c__shipout_vorigin_tl
```

Two constants holding the offset of the top-left with respect to the media box.

Setting the constants this way is courtesy of Bruno.

We delay setting the constants to the last possible place as there might be updates in the preamble or even in the `begindocument` hook that affects their setup.

```

308 \cs_new:Npn _shipout_init_page_origins: {
309 \tl_const:Nx \c__shipout_horigin_tl
310 {
311 \cs_if_exist_use:NTF \pdfvariable { horigin }
312 { \cs_if_exist_use:NF \pdfhorigin { 1in } }
313 }
314 \tl_const:Nx \c__shipout_vorigin_tl
315 {
316 \cs_if_exist_use:NTF \pdfvariable { vorigin }
317 { \cs_if_exist_use:NF \pdfvorigin { 1in } }
318 }
319 }
```

After the constants have been set there is no need to execute this command again, in fact it would raise an error, so we redefine it to do nothing.

```

319 \cs_gset_eq:NN _shipout_init_page_origins: \prg_do_nothing:
320 }
```

(End definition for `\_shipout_init_page_origins:, \c__shipout_horigin_tl, and \c__shipout_vorigin_tl.`)

`\_shipout_picture_overlay:n` Put the argument into a `picture` environment that doesn't take up any size and uses `1pt` for `\unitlength`.

*Todo: Could perhaps be generalized as it might be useful elsewhere. For now it is not.*

```

321 \cs_new:Npn _shipout_picture_overlay:n #1 {
```

The very first time this is executed we have to initialize (and freeze) the origins.

```

322 _shipout_init_page_origins:
```

```

323 \kern -\c__shipout_horigin_tl \scan_stop:
324 \vbox_to_zero:n {
325 \kern -\c__shipout_vorigin_tl \scan_stop:
326 \unitlength 1pt \scan_stop:

```

This mimics a simple zero-sized picture environment. The `\hss` is need in case there is horizontal material (without using `\put` with a positive width.

```

327 \hbox_set_to_wd:Nnn \l__shipout_tmp_box \c_zero_dim
328 { \ignorespaces #1 \hss }
329 \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
330 \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
331 \box_use:N \l__shipout_tmp_box
332 \tex_vss:D
333 }
334 }
```

(End definition for `\_shipout_picture_overlay:n`.)

`\_shipout_add_background_picture:n`

Put a `picture` env in the background of the shipout box with its reference point in the top-left corner.

```

335 \cs_new:Npn _shipout_add_background_picture:n #1 {
336 _shipout_add_background_box:n { _shipout_picture_overlay:n {#1} }
337 }
```

(End definition for `\_shipout_add_background_picture:n`.)

`\_shipout_add_foreground_picture:n`

Put a `picture` env in the foreground of the shipout box with its reference point in the top-left corner.

```

338 \cs_new:Npn _shipout_add_foreground_picture:n #1 {
339 _shipout_add_foreground_box:n { _shipout_picture_overlay:n {#1} }
340 }
```

(End definition for `\_shipout_add_foreground_picture:n`.)

`\shipout_discard:`

Request that the next shipout box should be discarded. At the moment this is just setting a boolean, but we may want to augment this behavior that the position of the call is taken into account (in case L<sup>A</sup>T<sub>E</sub>X looks ahead and is not using the position for on the next page).

```

341 \cs_new_protected:Npn \shipout_discard: {
342 \bool_gset_true:N \g__shipout_discard_bool
343 }
```

(End definition for `\shipout_discard::`. This function is documented on page 874.)

## 3.2 Handling the end of job hook

At the moment this is partly solved by using the existing hooks. But rather than putting the code into these hooks it should be moved to the right place directly as we shouldn't prefill hooks with material unless it needs to interact with other code.

`\g_shipout_READONLY_int`  
`\ ReadonlyShipoutCounter`

We count every shipout activity that makes a page (but not those that are discarded) in order to know how many pages got produced.

```

344 \int_new:N \g_shipout_READONLY_int
```

For L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub>  it is available as a command (i.e., a T<sub>E</sub>X counter only).

```
345 \cs_new_eq:NN \ ReadonlyShipoutCounter \g_shipout_READONLY_int
```

(End definition for `\g_shipout_READONLY_int` and `\ ReadonlyShipoutCounter`. These functions are documented on page 875.)

`\g_shipout_totalpages_int`  
`\c@totalpages`

We count every shipout attempt (even those that are discarded) in this counter. It is not used in the code but may get used in user code.

```
346 \int_new:N \g_shipout_totalpages_int
```

For L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub>  this is offered as a L<sup>A</sup>T<sub>E</sub>X counter so can be easily typeset inside the output routine to display things like “`\thepage/\thetotalpages`”, etc.

```
347 \cs_new_eq:NN \c@totalpages \g_shipout_totalpages_int
```

```
348 \cs_new:Npn \thetotalpages { \arabic{totalpages} }
```

(End definition for `\g_shipout_totalpages_int` and `\c@totalpages`. These functions are documented on page 875.)

`\@abspage@last`

In `\@abspage@last` record the number of pages from the last run. This is written to the `.aux` and this way made available to the next run. In case there is no `.aux` file or the statement is missing from it we initialize it with the largest possible number in T<sub>E</sub>X. We use this as the default because then we are inserting the `shipout/lastpage` on the last page (or after the last page) but not on page 1 for a multipage document.

```
349 \xdef\@abspage@last{\number\maxdimen}
```

(End definition for `\@abspage@last`.)

`\enddocument`

Instead of using the hooks `enddocument` and `enddocument/afterlastpage` we add this code to private kernel hooks to be 100% sure when it is executed and to avoid cluttering the hooks with data that is always there.

Inside `\enddocument` there is a `\clearpage`. Just before that we execute this code here. There is a good chance that we are on the last page. Therefore, if we don't know the value from the last run, we assume that the current page is the right one. So we set `\@abspage@last` and as a result the next shipout will run the `shipout/lastpage` code. Of course, if there are floats that still need a placement this guess will be wrong but then rerunning the document will give us the correct value next time around.

`\@kernel@after@enddocument`

```
350 \g@addto@macro \@kernel@after@enddocument {
351 \int_compare:nNnT \@abspage@last = \maxdimen
352 }
```

We use L<sup>A</sup>T<sub>E</sub>X coding as `\@abspage@last` is not an L3 name.

```
353 \xdef\@abspage@last{ \int_eval:n {\g_shipout_READONLY_int + 1} }
354 }
355 }
```

Once the `\clearpage` has done its work inside `\enddocument` we know for sure how many pages this document has, so we record that in the `.aux` file for the next run.

```
356 \g@addto@macro \@kernel@after@enddocument@afterlastpage {
```

There is one special case: If no output is produced then there is no point in a) recording the number as 0 will never match the page number of a real page and b) adding an extra page to ran the `shipout/lastpage` is pointless as well (as it would remain forever). So we test for this and run the code only if there have been pages.

```
357 \int_compare:nNnF \g_shipout_READONLY_int = 0
358 {
```

This ends up in the `.aux` so we use L<sup>A</sup>T<sub>E</sub>X 2<sub><</sub> names here.

*Todo: This needs an interface for `\nofiles` in `expl3`, doesn't at the moment!*

```
359 \if@filesw
360 \iow_now:Nx \@auxout {
361 \gdef\string\@abspage@last {\int_use:N \g_shipout_READONLY_int}
362 \fi
```

But we may have guessed wrongly earlier and we still have to run the `shipout/lastpage` even though there is no page to place it into. If that is the case we make a trivial extra page and put it there. This temporary page will then vanish again on the next run but helps to keep pdf viewers happy.

```
363 \bool_if:NF \g__shipout_lastpage_handled_bool
364 {
```

However, making this extra page in case the hook is actually empty would be forcing a rerun without any reason, so we check that condition and also check if `\@kernel@after@shipout@lastpage` contains any code. If both are empty we omit the page generation.

```
365 \bool_lazy_and:nnF
366 { \hook_if_empty_p:n {shipout/lastpage} }
367 { \tl_if_empty_p:N \@kernel@after@shipout@lastpage }
368 {
369 \tex_shipout:D\vbox to\textheight
370 {
371 \hbox:n { \UseHook{shipout/lastpage}
372 \@kernel@after@shipout@lastpage }
```

This extra page could be totally empty except for the hook content, but to help the user understanding why it is there we put some text into it.

```
373 __shipout_excuse_extra_page:
374 \null
375 }
```

At this point we also signal to L<sup>A</sup>T<sub>E</sub>X's endgame that a rerun is necessary so that an appropriate message can be shown on the terminal. We do this by simply defining a command used as a flag and tested in `\enddocument`.

```
376 \cs_gset_eq:NN \@extra@page@added \relax
377 }
378 }
379 }
380 }
```

(End definition for `\enddocument`, `\@kernel@after@enddocument`, and  
`\@kernel@after@enddocument@afterlastpage`.)

```

__shipout_excuse_extra_page: Say mea culpa ...
381 \cs_new:Npn __shipout_excuse_extra_page: {
382 \vfil
383 \begin{center}
384 \bfseries Temporary~ page!
385 \end{center}
386 \LaTeX{} was~ unable~ to~ guess~ the~ total~ number~ of~ pages~
387 correctly.~ ~ As~ there~ was~ some~ unprocessed~ data~ that~
388 should~ have~ been~ added~ to~ the~ final~ page~ this~ extra~
389 page~ has~ been~ added~ to~ receive~ it.
390 \par
391 If~ you~ rerun~ the~ document~ (without~ altering~ it)~ this~
392 surplus~ page~ will~ go~ away,~ because~ \LaTeX{} now~ knows~
393 how~ many~ pages~ to~ expect~ for~ this~ document.
394 \vfil
395 }

```

(End definition for `\_\_shipout\_excuse\_extra\_page:`.)

**\PreviousTotalPages** In the preamble before the `aux` file was read `\PreviousTotalPages` is always zero.

```
@kernel@before@begindocument \def\PreviousTotalPages{0}
```

In the `aux` file there should be an update for `\cabs@page@last` recording the number of pages from the previous run. If not that macro holds the value of `\maxdimen`. So we test for it and update `\PreviousTotalPages` if there was a real value. This should happen just before the `begindocument` hook is executed so that the value can be used inside that hook.

```
397 \g@addto@macro\@kernel@before@begindocument
398 {\ifnum\cabs@page@last<\maxdimen
399 \xdef\PreviousTotalPages{\cabs@page@last}\fi}
```

(End definition for `\PreviousTotalPages` and `\@kernel@before@begindocument`. These functions are documented on page 875.)

## 4 Legacy L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\epsilon$</sub> interfaces

**\DiscardShipoutBox** Request that the next shipout box is to be discarded.

```
400 \cs_new_eq:NN \DiscardShipoutBox \shipout_discard:
```

(End definition for `\DiscardShipoutBox`. This function is documented on page 874.)

**\AtBeginDvi** If we roll forward from an earlier kernel `\AtBeginDvi` is defined so we better not use `\cs_new_protected:Npn` here.

```
401 \cs_set_protected:Npn \AtBeginDvi
402 {__shipout_add_firstpage_material:Nn \AtBeginDvi}
```

(End definition for `\AtBeginDvi`. This function is documented on page 874.)

**\DebugShipoutsOn**

**\DebugShipoutsOff**

```
403 \cs_new_eq:NN \DebugShipoutsOn \shipout_debug_on:
404 \cs_new_eq:NN \DebugShipoutsOff \shipout_debug_off:
```

(End definition for `\DebugShipoutsOn` and `\DebugShipoutsOff`. These functions are documented on page 876.)

## 5 Internal commands needed elsewhere

These internal commands use double and triple @ signs so we need to stop getting them translated to the module name.

405 ⟨@@=⟩

Some internals needed elsewhere.

```
406 \cs_set_eq:NN \cexpl@@@shipout@add@firstpage@material@@Nn
407 __shipout_add_firstpage_material:Nn
408 \cs_set_eq:NN \cexpl@@@shipout@add@background@box@@n
409 __shipout_add_background_box:n
410 \cs_set_eq:NN \cexpl@@@shipout@add@foreground@box@@n
411 __shipout_add_foreground_box:n
412 \cs_set_eq:NN \cexpl@@@shipout@add@background@picture@@n
413 __shipout_add_background_picture:n
414 \cs_set_eq:NN \cexpl@@@shipout@add@foreground@picture@@n
415 __shipout_add_foreground_picture:n
```

(End definition for \cexpl@@@shipout@add@firstpage@material@@Nn and others.)

```
416 \ExplSyntaxOff
417 ⟨/2ekernel | latexrelease⟩
418 ⟨latexrelease⟩\EndIncludeInRelease
```

Rolling back here doesn't undefine the interface commands as they may be used in packages without rollback functionality. So we just make them do nothing which may or may not work depending on the code usage.

```
419 ⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
420 ⟨latexrelease⟩ {\shipout}{Hook management (shipout)}%
421 ⟨latexrelease⟩
```

If we roll forward then \tex\_shipout:D may not be defined in which case \shipout does have its original definition and so we must not \let it to something else which is \relax!

```
422 ⟨latexrelease⟩\ifcsname tex_shipout:D\endcsname
423 ⟨latexrelease⟩\expandafter\let\expandafter\shipout
424 ⟨latexrelease⟩ \csname tex_shipout:D\endcsname
425 ⟨latexrelease⟩\fi
426 ⟨latexrelease⟩
427 ⟨latexrelease⟩\let \RawShipout\@undefined
428 ⟨latexrelease⟩\let \ShipoutBox\@undefined
429 ⟨latexrelease⟩\let \ ReadonlyShipoutCounter \@undefined
430 ⟨latexrelease⟩\let \c@totalpages \@undefined
431 ⟨latexrelease⟩\let \thetotalpages \@undefined
432 ⟨latexrelease⟩
433 ⟨latexrelease⟩\let \DiscardShipoutBox \@undefined
434 ⟨latexrelease⟩\let \DebugShipoutsOn \@undefined
435 ⟨latexrelease⟩\let \DebugShipoutsOff \@undefined
436 ⟨latexrelease⟩
437 ⟨latexrelease⟩\DeclareRobustCommand \AtBeginDvi [1]{%
438 ⟨latexrelease⟩ \global \setbox \@begindvibox
439 ⟨latexrelease⟩ \vbox{\unvbox \@begindvibox #1}%
440 ⟨latexrelease⟩}
```

```

441 〈\latexrelease〉
442 〈\latexrelease〉\let \AtBeginShipout \@undefined
443 〈\latexrelease〉\let \AtBeginShipoutNext \@undefined
444 〈\latexrelease〉
445 〈\latexrelease〉\let \AtBeginShipoutFirst \@undefined
446 〈\latexrelease〉
447 〈\latexrelease〉\let \ShipoutBoxHeight \@undefined
448 〈\latexrelease〉\let \ShipoutBoxDepth \@undefined
449 〈\latexrelease〉\let \ShipoutBoxWidth \@undefined
450 〈\latexrelease〉

```

We do not undo a substitution when rolling back. As the file support gets undone the underlying data is no longer used (and sufficiently obscure that it should not interfere with existing commands) and properly removing it would mean we need to make the `\undeclare@...` and its support macros available in all earlier kernel releases which is pointless (and actually worse).

```

451 〈\latexrelease〉
452 〈\latexrelease〉\let \AtEndDvi \@undefined

```

We do not reenable a disabled package load when rolling back. As the file support gets undone the underlying data is no longer checked (and sufficiently obscure that it should not interfere with existing commands) and properly removing it would mean we need to make the `\reenable@package@load` command available in all earlier kernel releases which is pointless (and actually worse).

```

453 %\reenable@package@load{atenddvi}
454 〈\latexrelease〉
455 〈\latexrelease〉\EndIncludeInRelease
456 〈*2ekernel〉

```

## 6 Package emulation for compatibility

### 6.1 Package `atenddvi` emulation

**\AtEndDvi** This package has only one public command, so simulating it is easy and actually sensible to provide as part of the kernel.

```

457 〈/2ekernel〉
458 〈*2ekernel | latexrelease〉
459 〈\latexrelease〉\IncludeInRelease{2020/10/01}%
460 〈\latexrelease〉\AtEndDvi{atenddvi emulation}%
461 \ExplSyntaxOn
462 \cs_new_protected:Npn \AtEndDvi #1 {\AddToHook{shipout/lastpage}{#1}}
463 \ExplSyntaxOff

```

As the package is integrate we prevent loading (no need to roll that back):

```

464 \disable@package@load{atenddvi}
465 \PackageWarning{atenddvi}
466 {Functionality of this package is already\MessageBreak
467 provided by LaTeX.\MessageBreak\MessageBreak
468 It is there no longer necessary to load it.\MessageBreak
469 and you can safely remove it.\MessageBreak
470 Found on}%
471 〈/2ekernel | latexrelease〉

```

```

472 〈\latexrelease〉\EndIncludeInRelease
473 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
474 〈\latexrelease〉 {＼AtEndDvi}{atenddvi emulation}%
475 〈\latexrelease〉\let \AtEndDvi \undefined
476 〈\latexrelease〉\EndIncludeInRelease
477 〈*2ekernel〉

(End definition for \AtEndDvi. This function is documented on page 874.)
```

478 〈/2ekernel〉

## 6.2 Package **atbegshi** emulation

```

479 〈*atbegshi-ltx〉
480 \ProvidesPackage{atbegshi-ltx}
481 [2021/01/10 v1.0c
482 Emulation of the original atbegshi^~Jpackage with kernel methods]
```

**\AtBeginShipoutBox**

```
483 \let \AtBeginShipoutBox \ShipoutBox
```

(End definition for \AtBeginShipoutBox. This function is documented on page 876.)

**\AtBeginShipoutInit**

Compatibility only, we aren't delaying ...

```
484 \let \AtBeginShipoutInit \empty
```

(End definition for \AtBeginShipoutInit. This function is documented on page 877.)

**\AtBeginShipout**

**\AtBeginShipoutNext**

```

485 \protected\long\def\AtBeginShipout #1{\AddToHook{shipout/before}{#1}}
486 \protected\long\def\AtBeginShipoutNext #1{\AddToHookNext{shipout/before}{#1}}
```

(End definition for \AtBeginShipout and \AtBeginShipoutNext. These functions are documented on page 877.)

**\AtBeginShipoutFirst**

Slightly more complex as we need to know the name of the command under which the shipout/firstpage hook is filled.

```

487 \protected \def \AtBeginShipoutFirst
488 {\Expl@@@shipout@add@firstpage@material@Nn \AtBeginShipoutFirst}
```

(End definition for \AtBeginShipoutFirst. This function is documented on page 877.)

**\AtBeginShipoutDiscard**

Just a different name.

```
489 \let \AtBeginShipoutDiscard \DiscardShipoutBox
```

(End definition for \AtBeginShipoutDiscard. This function is documented on page 877.)

**\AtBeginShipoutAddToBox**

We don't expose them.

```
490 \let \AtBeginShipoutAddToBox
```

\Expl@@@shipout@add@background@box@n

**\AtBeginShipoutUpperLeft**

```
491 \let \AtBeginShipoutAddToBoxForeground
```

\Expl@@@shipout@add@foreground@box@n

**\AtBeginShipoutUpperLeftForeground**

```
492 \let \AtBeginShipoutUpperLeft
```

\Expl@@@shipout@add@background@picture@n

```
493 \let \AtBeginShipoutUpperLeftForeground
```

\Expl@@@shipout@add@foreground@picture@n

(End definition for `\AtBeginShipoutAddToBox` and others. These functions are documented on page 876.)

`\AtBeginShipoutOriginalShipout` This offers the raw `\shipout` primitive of the engine. A page shipped out with this is not counted by `\ReadonlyShipoutCounter` counter and thus the mechanism to place `\specials` at the very end of the output might fail, etc. It should therefore not be used in new applications but is only provided to allow running legacy code. For new code use the commands provided by the kernel instead.

```
498 \ExplSyntaxOn
499 \cs_new_eq:NN \AtBeginShipoutOriginalShipout \tex_shipout:D
```

(End definition for `\AtBeginShipoutOriginalShipout`. This function is documented on page 877.)

`\ShipoutBoxHeight` `\ShipoutBoxWidth` `\ShipoutBoxDepth` This is somewhat different from the original in `atbegshi` where `\ShipoutBoxHeight` etc. only holds the `\the\ht<box>` value. This may has some implications in some use cases and if that is a problem then it might need changing.

```
500 \cs_new:Npn \ShipoutBoxHeight { \dim_use:N \l_shipout_box_ht_dim }
501 \cs_new:Npn \ShipoutBoxDepth { \dim_use:N \l_shipout_box_dp_dim }
502 \cs_new:Npn \ShipoutBoxWidth { \dim_use:N \l_shipout_box_wd_dim }
503 \ExplSyntaxOff
```

(End definition for `\ShipoutBoxHeight`, `\ShipoutBoxWidth`, and `\ShipoutBoxDepth`.)

```
504 </atbegshi-ltx>
```

If the package is requested we substitute the one above:

```
505 <*2ekernel>
506 \declare@file@substitution{atbegshi.sty}{atbegshi-ltx.sty}
507 </2ekernel>
```

### 6.3 Package **everyshi** emulation

This is now directly handled in that package so emulation is not necessary any more.

Rather important :-)

```
508 <@@=
```

# File X

## ltoutput.dtx

### 1 Output Routine

#### 1.1 Floats

The ‘2ekernel’ code ensures that a `\usepackage{autoout1}` is essentially ignored if a ‘full’ format is being used that has the autoload file mode already in the format.

```
1 <defx>\begingroup
2 <defx>\makeatletter
3 <defx>\nfss@catcodes
4 <2ekernel>\expandafter\let\csname ver@autoout1.sty\endcsname\fmtversion
```

*Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*

```
5 <*2ekernel>
6 \message{output,}

* OUTPUT


```

#### PAGE LAYOUT PARAMETERS

```
\topmargin : Extra space added to top of page.
@twoside : boolean. T if two-sided printing
\oddsidemargin : IF @twoside = T
 THEN extra space added to left of odd-numbered
 pages.
 ELSE extra space added to left of all pages.
\evensidemargin : IF @twoside = T
 THEN extra space added to left of even-numbered
 pages.
\headheight : height of head
\headsep : separation between head and text
\footskip : distance separation between baseline of last
 line of text and baseline of foot.
 Note difference between \footSKIP and \headSEP.
\textheight : height of text on page, excluding head and foot
\textwidth : width of printing on page
\columnsep : IF @twocolumn = T
 THEN width of space between columns
\columnseprule : IF @twocolumn = T
 THEN width of rule between columns (0 if none).
\columnwidth : IF @twocolumn = T
 THEN ($\textwidth - \columnsep$)/2
 ELSE \textwidth
 It is set by the \twocolumn and
 \onecolumn commands.
```

- \@textbottom : Command executed at bottom of vbox holding text of page (including figures). The \raggedbottom command almost \let's this to \vfil (actually sets it to \vskip \z@ plus.0001fil). Should have depth 0pt.
- \@texttop : Command executed at top of vbox holding text of page (including figures). Used by letter style; can also be used to produce centered pages. Let to \relax by \raggedbottom and \flushbottom.

Page layout must initialize \@colht and \@colroom to \textheight.

#### PAGE STYLE PARAMETERS:

- \floatsep : Space left between floats.  
 \textfloatsep : Space between last top float or first bottom float and the text.  
 \topfigrule : Command to place rule (or whatever) between floats at top of page and text. Executed in inner vertical mode right before the \textfloatsep skip separating the floats from the text. Must occupy zero vertical space. (See \footnoterule.)  
 \botfigrule : Same as \topfigrule, but put after the \textfloatsep skip separating text from the floats at bottom of page.  
 \intextsep : Space left on top and bottom of an in-text float.  
 \dblfloatsep : Space between double-column floats.  
 \dbltextfloatsep : Space between top double-column floats and text.  
 \dblfigrule : Similar to \topfigrule, but for double-column floats.  
 \@fptop : Glue to go at top of float column – must be 0pt + stretch  
 \@fpsep : Glue to go between floats in a float column.  
 \@fpbot : Glue to go at bottom of float column – must be 0pt + stretch  
 \@dblfpsep, \@dblfpbot : Analogous for double-column float page in two-column format.

FOOTNOTES: As in PLAIN, footnotes use \insert\footins.

#### PAGE LAYOUT SWITCHES AND MACROS

- @twocolumn : Boolean. T if two columns per page globally.

#### PAGE STYLE MACROS AND SWITCHES

```

\@oddhead : IF @twoside = T
 THEN macro to generate head of odd-numbered
 pages.
 ELSE macro to generate head of all pages.
\@evenhead : IF @twoside = T
 THEN macro to generate head of even-numbered
 pages.
\@oddfoot : IF @twoside = T
 THEN macro to generate foot of odd-numbered
 pages.
 ELSE macro to generate foot of all pages.
\@evenfoot : IF @twoside = T
 THEN macro to generate foot of even-numbered
 pages.
@specialpage : boolean. T if current page is to have a special
 format.
\@specialstyle : If its value is foo then
 IF @specialpage = T
 THEN the command \ps@foo is executed to
 temporarily reset the page style parameters
 before composing the current page.
 This command should execute only \def's and
 \edef's, making only local definitions.

```

## FLOAT PLACEMENT PARAMETERS

The following parameters are set by the macro `\@floatplacement`.

When `\@floatplacement` is called,

`\@colht` is the height of the page or column being built. I.e.:

- \* For single-column page it equals `\textheight`.
- \* For double-column page it equals `\textheight - height` of double-column floats on page.

Note that some are set globally and some locally:

```

\@topnum :=G Maximum number of floats allowed on the top of a
 column.
\@toproom :=G Maximum amount of top of column devoted to floats-
 excluding \textfloatsep separation below the floats
 and \floatsep separation between them. For
 two-column output, should be computed as a function
 of \@colht.
\@botnum, \@botroom
 : Analogous to above.
\@colnum :=G Maximum number of floats allowed in a column,
 including in-text floats.
\@textmin :=L Minimum amount of text (excluding footnotes) that
 must appear on a text page.
 %% 27 Sep 85 : made local to
 %% \@addtocurcol and \@addtonextcol
 It is now also used locally in processing double
 floats.

```

`\@fpmin :=L` Minimum height of floats in a float column.

The macro `\@dblfloatplacement` sets the following parameters.

`\@dbltopnum :=G` Maximum number of double-column floats allowed at the top of a two-column page.

`\@dbltoproom :=G` Maximum height of double-column floats allowed at top of two-column page.

`\@fpmin :=L` Minimum height of floats in a float column.

It should also perform the following local assignments where necessary – i.e., where the new value differs from the old one:

`\@fptop :=L \@dblftop`

`\@fpsep :=L \@dblfpsep`

`\@fpbot :=L \@dblfpbot`

## OUTPUT ROUTINE VARIABLES

`\@colht` : The total height of the current column. In single column style, it equals `\textheight`. In two-column style, it is `\textheight` minus the height of the double-column floats on the current page. MUST BE INITIALIZED TO `\textheight`.

`\@colroom` : The height available in the current column for text and footnotes. It equals `\@colht` minus the height of all floats committed to the top and bottom of the current column.

`\@textfloatsheight` : The total height of in-text floats on the current page.

`\footins` : Footnote insertion number.

`\@maxdepth` : Saved value of TeX's `\maxdepth`. Must be set when any routine sets `\maxdepth`.

## CALLING THE OUTPUT ROUTINE

---

The output routine is called either by TeX's normal page-breaking mechanism, or by a macro putting a penalty < or = -10000 in the output list. In the latter case, the penalty indicates why the output routine was called, using the following code.

| penalty | reason                                                            |
|---------|-------------------------------------------------------------------|
| -10000  | <code>\pagebreak</code><br><code>\newpage</code>                  |
| -10001  | <code>\clearpage (\penalty -10000 \vbox{} \penalty -10001)</code> |
| -10002  | float insertion, called from horizontal mode                      |
| -10003  | float insertion, called from vertical mode.                       |
| -10004  | float insertion.                                                  |

Note: A float or marginpar puts the following sequence in the output list:

- (i) a penalty of -10004,
- (ii) a null \vbox
- (iii) a penalty of -10002 or -10003.

This solves two special problems:

1. If the float comes right after a \newpage or \clearpage, then the first penalty is ignored, but the second one invokes the output routine.
2. If there is a split footnote on the page, the second 'page' puts out the rest of the footnote.

## THE OUTPUT ROUTINE

---

### FUNCTIONS USED IN THE OUTPUT ROUTINE:

\@outputpage : Produces an output page with the contents of box \@outputbox as the text part.

Also sets \@colht :=G \textheight.

The page style is determined as follows.

IF @thispagestyle = true  
THEN use \thispagestyle style  
ELSE use ordinary page style.

\@tryfcolumn\FLIST : Tries to form a float column composed of floats from \FLIST (if nonempty) with the following parameters:

\@colht : height of box  
\@fpmmin : minimum height of floats in the box  
\@fpsep : interfloat space  
\@ftpsep : glue at top of box  
\@fpbot : glue at bottom of box.

If it succeeds, then it does the following:

\* \@outputbox :=L the composed float box.  
\* @fcollmade :=G true  
\* \FLIST :=G \FLIST - floats put in box  
\* \@freelist :=G \@freelist + floats put in box

If it fails, then:

\* @fcollmade :=G false

NOTE: BIT MUST BE A SINGLE TOKEN!

\@makefcolumn \FLIST : Same as \@tryfcolumn except that it fails to make a float column only if \FLIST is empty. Otherwise, it makes a float column containing at least the first box in \FLIST, disregarding \@fpmmin.

\@startcolumn :

Calls \@tryfcolumn@\deferlist. If \@tryfcolumn returns with (globally set) @fcollmade = false, then:

\* Globally sets \@toplist and \@botlist to floats

from `\@deferlist` to go at top and bottom of column, deleting them from `\@deferlist`. It does this using `\@colht` as the total height, the page style parameters `\@floatsep` and `\@textfloatsep`, and the float placement parameters `\@topnum`, `\@toproom`, `\@botnum`, `\@botroom`, `\@colnum` and `\textfraction`.

- \* Globally sets `\@colroom` to `\@colht` minus the height of the added floats.

`\@startdblcolumn` :

Calls `\@tryfcolumn\@dbldeferlist{8}`. If `\@tryfcolumn` returns with (globally set) `@fcolmade = false`, then:

- \* Globally sets `\@dbltopl` to floats from `\@dbldeferlist` to go at top and bottom of column, deleting them from `\@dbldeferlist`. It does this using `\textheight` as the total height, and the parameters `\@dblfloatsep`, etc.
- \* Globally sets `\@colht` to `\textheight` minus the height of the added floats.

`\@combinefloats` : Combines the text from box

`\@outputbox` with the floats from `\@topl` and `\@botl`, putting the new box in `\@outputbox`. It uses `\floatsep` and `\textfloatsep` for the appropriate separations. It puts the elements of `\TOPLIST` and `\BOTLIST` onto `\@freelist`, and makes those lists null.

`\@makecol` : Makes the contents of `\box255` plus the accumulated footnotes, plus the floats in `\@topl` and `\@botl`, into a single column of height `\@colht` (unless the page height has been locally changed), which it puts into box `\@outputbox`. It puts boxes in `\@midlist` back onto `\@freelist` and restores `\maxdepth`.

`\@opcol` : Outputs a column whose text is in box `\@outputbox`

If `@twocolumn = false`, then it calls `\@outputpage`, sets `\@colht :=G \textheight`, and calls `\@floatplacement`.

If `@twocolumn = true`, then:

If `@firstcolumn = true`, then it puts box `\@outputbox` into `\@leftcolumn` and sets `@firstcolumn :=G false`.

If `@firstcolumn = false`, then it puts out the current two-column page, any possible two-column float pages, and determines `\@dbltopl` for the next page.

## USER COMMANDS THAT CALL OR AFFECT THE OUTPUT ROUTINE

---

```

\newpage == BEGIN \par\vfil\penalty -10000 END

\clearpage == BEGIN \newpage
 \write -1{} % Part of hack to make sure no
 \vbox{} % \write's get lost.
 \penalty -10001
END

\cleardoublepage == BEGIN \clearpage
 if @twoside = true and c@page is even
 then \hbox{} \newpage fi
 END

```

\twocolumn[BOX] : starts a new page, changing to twocolumn setting and puts BOX in a parbox of width \textwidth across the top. Useful for full-width titles for double-column pages.  
SURPRISE: The stretch from \dbltextfloatsep will be inserted between the BOX and the top of the two columns.

## FLOAT-HANDLING MECHANISMS

---

The float environment obtains an insertion number B from the \freelist (see below for a description of list manipulation), puts the float into box B and sets \count B to a FLOAT SPECIFIER. For a normal (not double-column) float, it then causes a page break in one of the following two ways:

- In outer hmode: \vadjust{\penalty -10002}
- In vmode : \penalty -10003.

For a double-column float, it puts B onto the \dbldeferlist.

The float specifier has two components:

- \* A PLACEMENT SPECIFICATION, describing where the float may be placed.
- \* A TYPE, which is a power of two—e.g., figures might be type 1 floats, tables type 2 floats, programs type 4 floats, etc.

The float specifier is encoded as follows, where bit 0 is the least significant bit.

| Bit | Meaning                                              |
|-----|------------------------------------------------------|
| 0   | 1 iff the float may go where it appears in the text. |
| 1   | 1 iff the float may go on the top of a page.         |
| 2   | 1 iff the float may go on the bottom of a page.      |
| 3   | 1 iff the float may go on a float page.              |
| 4   | 1 unless the PLACEMENT includes a !                  |
| 5   | 1 iff a type 1 float                                 |

6        1 iff a type 2 float  
etc.

A negative float specifier is used to indicate a marginal note.

## MACROS AND DATA STRUCTURES FOR PROCESSING FLOATS

---

A FLOAT LIST consisting of the floats in boxes `\boxa` ... `\boxN` has the form:

`\@elt \boxa ... \@elt \boxN`

where `\boxI` is defined by

`\newinsert\boxI`

Normally, `\@elt` is `\let` to `\relax`. A test can be performed on the entire float list by locally `\def`'ing `\@elt` appropriately and executing the list.

This is a lot more efficient than looping through the list.

The following macros are used for manipulating float lists.

```
\@next \CS \LIST {\NONEMPTY}{\EMPTY} == %% NOTE: ASSUME \@elt
= \relax
 BEGIN assume that \LIST == \@elt \B1 ... \@elt \Bn
 if n = 0
 then EMPTY
 else \CS :=L \B1
 \LIST :=G \@elt \B2 ... \@elt \Bn
 NONEMPTY
 fi
 END
```

`\@bitor\NUM\LIST` : Globally sets switch `@test` to the disjunction for all I of bit  $\log_2 \NUM$  of the float specifiers of all the floats in `\LIST`.  
I.e., `@test` is set to true iff there is at least one float in `\LIST` having bit  $\log_2 \NUM$  of its float specifier equal to 1.

Note:  $\log_2 [(\count I)/32]$  is the bit number corresponding to the type of float I. To see if there is any float in `\LIST` having the same type as float I, you run `\@bitor` with

`\NUM = [(\count I)/32] * 32.`

```
\@bitor\NUM\LIST ==
BEGIN
 @test :=G false
 { \@elt \CTR == if \NUM <> 0 then
 if \count\CTR / \NUM is odd
 then @test := true fi fi
```

```

 \LIST
}
END

```

\@cons\LIST\NUM : Globally sets \LIST := \LIST \* \@elt \NUM

```

\@cons\LIST\NUM ==
BEGIN { \@elt == \relax
 \LIST :=G \LIST \@elt \NUM
}

```

## BOX LISTS FOR FLOAT-PLACEMENT ALGORITHMS

|                |                                                            |
|----------------|------------------------------------------------------------|
| \@freelist     | : List of empty boxes for placing new floats.              |
| \@toplist      | : List of floats to go at top of current column.           |
| \@midlist      | : List of floats in middle of current column.              |
| \@botlist      | : List of floats to go at bottom of current column.        |
| \@deferlist    | : List of floats to go after current column.               |
| \@dbltoplist   | : List of double-col. floats to go at top of current page. |
| \@dbldeferlist | : List of double-column floats to go on subsequent pages.  |

## FLOAT-PLACEMENT ALGORITHMS

\@addtobot : Tries to put insert \@currbox on \@botlist.

Called only when:

- \* \ht BOX < \@colroom
- \* type of \@currbox not on \@deferlist
- \* \@colnum > 0
- \* @insert = false

If it succeeds, then:

- \* sets @insert true
- \* decrements \@botroom by \ht BOX
- \* decrements \@botnum and \@colnum by 1
- \* decrements \@colroom by \ht BOX + either \floatsep or \textfloatsep, as appropriate.
- \* sets \maxdepth to 0pt

\@addtotoporbot : Tries to put insert \@currbox on \@toplist or \@botlist.

Called only under same conditions as \@addtobot.

If it succeeds, then:

- \* sets @insert true
- \* decrements \@toproom or \@botroom by \ht BOX
- \* decrements \@colnum and either \@topnum or \@botnum by 1
- \* decrements \@colroom by \ht BOX + \floatsep

or `\textfloatsep`, as appropriate.

`\@addtocurcol` : Tries to add `\@currbox` to current column, setting  
  `@insert` true if it succeeds, false otherwise.  
  It will add `\@currbox` to top only if bit 0 of  
  `\count\@currbox` is 0, and to the bottom only if  
  bit 0 = 0 or an earlier float of the same type is  
  put on the bottom.  
  If the float is put in the text, then  
  `\penalty\interlinepenalty` is put  
  right after the float, before the following `\vskip`,  
  and `\outputpenalty :=L 0`.

`\@addtonextcol` : Tries to add `\@currbox` to the next column, setting  
  `@insert` true if it succeeds, false otherwise.

`\@addtobdblcol` : Tries to add `\@currbox` to the next double-column page,  
  adding it to `\@dbltoplist` if it succeeds and  
  `\@dbldefeolist` if it fails.

```
\@addmarginpar ==
BEGIN
if \@currlist nonempty
 then remove \@marbox from \@currlist
 add \@marbox and \@currbox to \@freelist
 %% NOTE: \@currbox = left box
 else LaTeX error: ? %% shouldn't happen
fi
\@tempcnta := 1 %% 1 = right, -1 = left
if @twocolumn = true
 then if @firstcolumn = true
 then \@tempcnta := -1
 fi
 else if @mparswitch = true
 then if count0 odd
 else \@tempcnta := -1
 fi
 fi
 if @reversemargin = true
 then \@tempcnta := -\@tempcnta
 fi
fi
if \@tempcnta < 0 then \box\@marbox :=G \box\@currbox
fi
\@tempdima :=L maximum(\@mparbottom - \@pageht
 + ht of \@marbox, 0)
if \@tempdima > 0 then LaTeX warning: 'marginpar moved' fi
\@mparbottom :=G \@pageht + \@tempdima + depth of \@marbox
 + \marginpush
```

```

\@tempdima :=L \@tempdima - ht of \@marbox
\box\@marbox :=G \box\@currbox
 \vbox { \vskip \@tempdima
 \box\@marbox
 }
height of \@marbox :=G depth of \@marbox :=G 0
\kern -\@pagedp
\nointerlineskip
\hbox{ if @tempcpta > 0 then \hskip \columnwidth
 \hskip \marginparsep
 else \hskip -\marginparsep
 \hskip -\marginparwidth
 fi
 \box\@marbox \hss
}
\nobreak
\nointerlineskip
\hbox{\vrule height 0 width 0 depth \@pagedp}
END

```

FLOATS AND MARGINPARS ADD A LOT OF DEAD CYCLES.

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

```

7 \maxdeadcycles = 100
8 \let\@elt\relax
9 \def\@next#1#2#3#4{\ifx#2\empty #4\else
10 \expandafter\@next #2\@#1#2#3\fi}
11 \def\@xnext \@elt #1#2\@#3#4{\def#3{#1}\gdef#4{#2}}
12 \def\@testfalse{\global\let\if@test\iffalse}
13 \def\@testtrue {\global\let\if@test\iftrue}
14 \qquad\@testfalse
15 \def\@bitor#1#2{\@testfalse {\let\@elt\@xbitor
16 \@tempcpta #1\relax #2}}

```

RmS 91/11/22: Added test for \count#1 = 0. Suggested by Chris Rowley.

```

17 \def\@xbitor #1{\@tempcntb \count#1
18 \ifnum \@tempcpta =\z@
19 \else
20 \divide\@tempcntb\@tempcpta
21 \ifodd\@tempcntb \@testtrue\fi
22 \fi}

```

#### DEFINITION OF FLOAT BOXES:

```

23 </2ekernel>
24 <|latexrelease>\IncludeInRelease{2015/10/01}%
25 <|latexrelease> \bx@ZZ-{Extended float list}%
26 <*2ekernel | latexrelease>
27 \let\@elt\newinsert
28 <*2ekernel>
29 \def\@freelist{%

```

```

30 \@elt\bx@A\@elt\bx@B\@elt\bx@C\@elt\bx@D\@elt\bx@E
31 \@elt\bx@F\@elt\bx@G\@elt\bx@H\@elt\bx@I\@elt\bx@J
32 \@elt\bx@K\@elt\bx@L\@elt\bx@M\@elt\bx@N
33 \@elt\bx@O\@elt\bx@P\@elt\bx@Q\@elt\bx@R}
34 \@freelist
35 </2ekernel>
36 \ifx\numexpr\@undefined\else
37 \def\reserved@a{%
38 \@elt\bx@S\@elt\bx@T\@elt\bx@U\@elt\bx@V
39 \@elt\bx@W\@elt\bx@X\@elt\bx@Y\@elt\bx@Z
40 \@elt\bx@AA\@elt\bx@BB\@elt\bx@CC\@elt\bx@DD\@elt\bx@EE
41 \@elt\bx@FF\@elt\bx@GG\@elt\bx@HH\@elt\bx@II\@elt\bx@JJ
42 \@elt\bx@KK\@elt\bx@LL\@elt\bx@MM\@elt\bx@NN
43 \@elt\bx@OO\@elt\bx@PP\@elt\bx@QQ\@elt\bx@RR
44 \@elt\bx@SS\@elt\bx@TT\@elt\bx@UU\@elt\bx@VV
45 \@elt\bx@WW\@elt\bx@XX\@elt\bx@YY\@elt\bx@ZZ}
46 \reserved@a
47 \def\@elt{\noexpand\@elt\noexpand}
48 \edef\@freelist{\@freelist\reserved@a}
49 \fi
50 \let\reserved@a\relax
51 \let\@elt\relax
52 </2ekernel | latexrelease>
53 <latexrelease>\EndIncludeInRelease
54 <latexrelease>\IncludeInRelease{0000/00/00}%
55 <latexrelease> {\bx@ZZ}{Extended float list}%
56 <latexrelease>\def\@freelist{%
57 <latexrelease> \@elt\bx@A\@elt\bx@B\@elt\bx@C\@elt\bx@D\@elt\bx@E
58 <latexrelease> \@elt\bx@F\@elt\bx@G\@elt\bx@H\@elt\bx@I\@elt\bx@J
59 <latexrelease> \@elt\bx@K\@elt\bx@L\@elt\bx@M\@elt\bx@N
60 <latexrelease> \@elt\bx@O\@elt\bx@P\@elt\bx@Q\@elt\bx@R}
61 <latexrelease> \insc@unt=234
62 <latexrelease>\EndIncludeInRelease
63 <*2ekernel>

64 \gdef\@toplist{}
65 \gdef\@botlist{}
66 \gdef\@midlist{}
67 \gdef\@currlist{}
68 \gdef\@deferlist{}
69 \gdef\@dbltoplist{}

```

The new algorithm stores page wide floats together with column floats in a single `\@deferlist` list. We keep `\@dbldeferlist` initialised as empty so that packages that are testing for deferred floats can use the same code for old or new float handling.

```

70 \gdef\@dbldeferlist{}
 PAGE LAYOUT PARAMETERS
71 \newdimen\topmargin
72 \newdimen\oddsidemargin
73 \newdimen\evensidemargin
74 \let\@themargin=\oddsidemargin
75 \newdimen\headheight
76 \newdimen\headsep
77 \newdimen\footskip

```

```

78 \newdimen\textheight
79 \newdimen\textwidth
80 \newdimen\columnwidth
81 \newdimen\columnsep
82 \newdimen\columnseprule
83 \newdimen\marginparwidth
84 \newdimen\marginparsep
85 \newdimen\marginparpush

```

**\AtBeginDvi** We use a box register in which to put stuff that must appear before anything else in the .dvi file.

The stuff in the box should not add any typeset material to the page when it is unboxed.

This interface is no longer used. Instead a new one is inside `ltshipout.dtx`. We only keep the box in case some old code refers to it directly (or we do some rollback).

```

86 \newbox\@begindvibox
87 %\DeclareRobustCommand \AtBeginDvi [1]{%
88 % \global \setbox \@begindvibox
89 % \vbox{\unvbox \@begindvibox #1}%
90 %}

```

(End definition for `\AtBeginDvi` and `\@begindvibox`. These functions are documented on page 874.)

**\@maxdepth** This is not the right place to set this; it needs to be set in a class/style file when `\maxdepth` is set.

Also, many settings to `\maxdepth` should be to `\@maxdepth`, probably?

```

91 \newdimen\@maxdepth
92 \@maxdepth = \maxdepth

```

(End definition for `\@maxdepth`.)

**\paperheight** New `\paper...` registers.

```

93 \newdimen\paperheight
94 \newdimen\paperwidth

```

(End definition for `\paperheight` and `\paperwidth`.)

**\if@insert** Local switches first:

```

95 \newif \if@insert

```

**\if@fcolmade** These should definitely be global:

```

96 \newif \if@fcolmade
97 \newif \if@specialpage \@specialpagefalse

```

**\if@reversemarginpar** These should be global but are not always set globally in other files.

```

98 \newif \if@firstcolumn \@firstcolumntrue
99 \newif \if@twocolumn \@twocolumnfalse

```

Not sure about these: two questions. Should things which must apply to a whole document be local or global (they probably should be ‘preamble only’ commands)? Are these three such things?

```

100 \newif \if@twoside \@twosidefalse
101 \newif \if@reversemargin \@reversemarginfalse
102 \newif \if@mparswitch \@mparswitchfalse

```

This counter has been imported from ‘multicol’.

```
103 \newcount \col@number
104 \col@number \cne
```

(End definition for `\if@insert` and others.)

Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):

#### INTERNAL REGISTERS

```
105 \newcount\@topnum
106 \newdimen\@toproom
107 \newcount\@dbltopnum
108 \newdimen\@dbltoproom
109 \newcount\@botnum
110 \newdimen\@botroom
111 \newcount\@colnum
112 \newdimen\@textmin
113 \newdimen\@fpmin
114 \newdimen\@colht
115 \newdimen\@colroom
116 \newdimen\@pageht
117 \newdimen\@pagedp
118 \newdimen\@mparbottom \cne \z@
119 \newcount\@currtype
120 \newbox\@outputbox
121 \newbox\@leftcolumn
122 \newbox\@holdpg
```

```
123 \def\@thehead{\@oddhead} % initialization
124 \def\@thefoot{\@oddfoot}
```

End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.

- `\clearpage` The tests at the beginning are an experimental attempt to avoid a completely empty page after a `\twocolumn[...]`. This prevents the text from the argument vanishing into a float box, never to be seen again. We hope that it does not produce wrong formatting in other cases.

```
125 \def\clearpage{
126 \ifvmode
127 \ifnum \@dbltopnum =\cne
128 \ifdim \pagetotal <\z@
129 \hbox{}%
130 \fi
131 \fi
132 \fi
133 \newpage
134 \write\m@ne{}%
135 \vbox{}%
136 \penalty -\z@
137 }
```

(End definition for `\clearpage`.)

```

\cleardoublepage
138 \def\cleardoublepage{\clearpage\if@twoside \ifodd\c@page\else
139 \hbox{}\newpage\if@twocolumn\hbox{}\newpage\fi\fi\fi}
140
```

(End definition for `\cleardoublepage`.)

```

\onecolumn
141 <*2ekernel | fltrace>
142 \def\onecolumn{%
143 \clearpage
144 \global\columnwidth\textwidth
145 \global\hsize\columnwidth
146 \global\linewidth\columnwidth
147 \global\@twocolumnfalse
148 \col@number \@ne
149 \@floatplacement}

```

(End definition for `\onecolumn`.)

`\newpage` The two checks at the beginning ensure that an item label or run-in section title immediately before a `\newpage` get printed on the correct page, the one before the page break.

All three tests are largely to make error processing more robust; that is why they all reset the flags explicitly, even when it would appear that this would be done by a `\leavevmode`.

```

150 </2ekernel | fltrace>
151 <latexrelease>\IncludeInRelease{2017/04/15}%
152 <latexrelease> {\newpage}{Check depth of page}%
153 <*2ekernel | latexrelease | fltrace>
154 \def \newpage {%
155 \if@noskipsec
156 \ifx \@nodocument\relax
157 \leavevmode
158 \global \@noskipsecfalse
159 \fi
160 \fi
161 \if@inlabel
162 \leavevmode
163 \global \@inlabelfalse
164 \fi
165 \if@nobreak \@nobreakfalse \everypar{} \fi
166 \par

```

The `\vfil` at the end of the macro before the break penalty will normally result in the page being run short, even with `\flushbottom` in effect (in contrast to the behavior of `\pagebreak`). However, if there is some explicit stretch on the page, say, a `\vfill`, it has the undesired side-effect, that the last line will not align at its baseline if it contains characters going below the baseline, as the value of `\prevdepth` is no longer taken into account by TeX. So we back up by that amount (or by `\maxdepth` if it is really huge), to mimic the normal behavior without the `\newpage`.

```

167 \ifdim\prevdepth>\z@
168 \vskip -%
169 \ifdim\prevdepth>\maxdepth

```

```

170 \maxdepth
171 \else
172 \prevdepth
173 \fi
174 \fi
175 \vfil
176 \penalty -\@M}
177 </2ekernel | latexrelease | fltrace>
178 <latexrelease>\EndIncludeInRelease
179 <latexrelease>\IncludeInRelease{0000/00/00}%
180 <latexrelease> {\newpage}{Check depth of page}%
181 <latexrelease>\def \newpage {%
182 <latexrelease> \if@noskipsec
183 <latexrelease> \ifx \nодокумент\relax
184 <latexrelease> \leavevmode
185 <latexrelease> \global \nокомпенсацияfalse
186 <latexrelease> \fi
187 <latexrelease> \fi
188 <latexrelease> \if@inlabel
189 <latexrelease> \leavevmode
190 <latexrelease> \global \инлайнfalse
191 <latexrelease> \fi
192 <latexrelease> \if@nobreak \nobreakfalse \everypar{} \fi
193 <latexrelease> \par
194 <latexrelease> \vfil
195 <latexrelease> \penalty -\@M}
196 <latexrelease>\EndIncludeInRelease
197 <*2ekernel | fltrace>

```

(*End definition for \newpage.*)

\@emptycol It may be better to use an invisible rule rather than an empty box here.

```
198 \def \@emptycol {\vbox{} \penalty -\@M}
```

(*End definition for \@emptycol.*)

\twocolumn There are several bug fixes to the two-column stuff here.

```

199 \def \twocolumn {%
200 \clearpage
201 \global \columnwidth \textwidth
202 \global \advance \columnwidth -\columnsep
203 \global \divide \columnwidth \tw@
204 \global \hsize \columnwidth
205 \global \linewidth \columnwidth
206 \global \twocolumntrue
207 \global \firstcolumntrue
208 \col@number \tw@

```

There is no reason to put a \@dblfloatplacement here since \@topnewpage ignores these settings. The \@floatplacement is needed in case this comes after some changes.

```

209 \@ifnextchar [\@topnewpage \@floatplacement
210 }
```

Note that here, getting a box from the freelist can assume success since this comes just after a `\clearpage`.

```

211 \long\def \@topnewpage [#1]{%
212 \@nodocument
213 \@next\@currbox\@freelist{}{}%
214 \global \setbox\@currbox
215 \color@vbox
216 \normalcolor
217 \vbox {%
218 \hsize\textwidth
219 \parboxrestore
220 \col@number \@ne
221 #1%
222 \vskip -\dbltextfloatsep
223 }%
224 \color@endbox

```

Added size test and warning message; perhaps we should use an error message.

```

225 \ifdim \ht\@currbox>\textheight
226 \ht\@currbox \textheight
227 \fi

```

This next line is not essential but it is more robust to make this value non-zero, in case of weird errors.

This next bit is what is needed from `\@addtobblcol`, plus some extra checks for error trapping.

```

228 \global \count\@currbox \tw@
229 \tempdima -\ht\@currbox
230 \advance \tempdima -\dbltextfloatsep
231 \global \advance \colht \tempdima
232 \ifx \dbltoplist \empty
233 \else
234 \@latex@error{Float(s) lost}\ehb
235 \let \dbltoplist \empty
236 \fi
237 \cons \dbltoplist \@currbox

```

This setting of `\@dbltopnum` is used only to change the typesetting in `\@combinedblfloats`.

```

238 \global \@dbltopnum \m@ne
239 <*trace>
240 \f@trace{dbltopnum set to -1 (= \the \@dbltopnum) (topnewpage)}%
241 />trace

```

At points such as this we need to check that there is still a minimal amount of room left on the page; this uses an arbitrary small value at present; but note that this value is larger than that used when checking that page is too full of normal floats.

If there is little room left we just force a page-break, OK? This involves producing two empty columns. The second empty column may be produced by `\output`, in which case an extra, misleading, warning will be generated, OK? (This happens only when there is too little room left on the page for any float.) Otherwise (i.e. if the size is such that it is allowed as a normal float) the extra `\@emptycol` will be invoked in the second column by the conditional code guarded by the `\if@firstcolumn` test.

I now think that the cut-off point here should be  $3\baselineskip$ , but we make it a bit less so that 3 lines of text will be allowed, OK?

Since this happens only when there is nothing on the page but the ‘top-box’, the empty box should not cause any problem other than some overfull box messages, which is not entirely misleading.

Here we need two page-ends since both columns need to be empty.

```

242 \ifdim \@colht<2.5\baselineskip
243 \@latex@warning@no@line {Optional argument of \noexpand\twocolumn
244 too tall on page \thepage}%
245 \emptycol
246 \if@firstcolumn
247 \else
248 \emptycol
249 \fi
250 \else
251 \global \vsize \@colht
252 \global \@colroom \@colht
253 \@floatplacement
254 \fi
255 }
```

(End definition for `\twocolumn` and `\@topnewpage`.)

`\output` This needs some small adjustments. We cannot guarantee that the float mechanism will interact correctly with this stuff, but that mechanism does not always work properly with footnotes already.

RmS 91/09/29:

added reset of `\par` to the output routine. This avoids problems when the output routine is called within a list where `\par` may be a no-op.

```

256 \output {%
257 \let \par \@@par
258 \ifnum \outputpenalty<-\@M
259 \specialoutput
260 \else
261 \makecol
262 \opcol
```

Moved to `\opcol`: `\@floatplacement`.

```
263 \startcolumn
```

This loop could be replaced by an `\expandafter` tail recursion in `\startcolumn`.

```

264 \whilesw \if@fcolmade \fi
265 {%
266 (*trace)
267 \f@trace{PAGE: float \if@twocolumn column \else page \fi
268 completed}%
269 (/trace)
270 \opcol\startcolumn}%
271 \fi
272 \ifnum \outputpenalty>-\@Miv
```

At points such as this we need to check that there is still a minimal amount of room left on the page; this uses an arbitrary small value at present. If there is little room left we just force a page-break, OK?

This bit is essential only if a float has just been processed so maybe it should be moved; but this is the natural place at which to set the vsize and a test would need to be done anyway. A check has been added to ensure that there really has been a change in the value of \@colroom.

Since this happens only when there is nothing on the page but floats, the empty box should not cause any problem other than some overfull box messages, which is not entirely misleading.

The twocolumn case does not need any extra code here since this is the \output itself; in the second column there will still not be enough room left so \@emptycol will be executed again when the OR is called by the-page builder when it gets to the penalty inserted by the first execution. (The page-builder is never invoked whilst the OR is being executed since it builds a inner vlist; thus any conditional code for the two-column case within \output may not get executed with the correct value of \if@firstcolumn.

```

273 \ifdim \@colroom<1.5\baselineskip
274 \ifdim \@colroom<\textheight
275 \@latex@warning@no@line {Text page \thepage\space
276 contains only floats}%
277 \@emptycol
278 %
279 \if@twocolumn
280 %
281 \else
282 %
283 \fi
284 %
285 \global \vsize \@colroom
286 \fi
287 \else
288 \global \vsize \@colroom
289 \fi
290 \else
291 \global \vsize \maxdimen
292 \fi
293 }
```

*Historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments (not necessarily accurate any more):*

CHANGES TO \@specialoutput:

\* \penalty\z@ changed to \penalty\interlinepenalty so \samepage works properly with figure and table environments.

(Changed 23 Oct 86)

\* Definition of \@specialoutput changed 26 Feb 88 so \@pageht and \@pagedp aren't changed for a marginal note.

(Change suggested by Chris Rowley.)

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

```

294 \gdef\@specialoutput{%
295 \ifnum \outputpenalty>-\@Mii
296 \@doclearpage
```

```

297 \else
298 \ifnum \outputpenalty<-\@Mii
299 \ifnum \outputpenalty<-\@MM \deadcycles \z@ \fi
300 \global \setbox\@holdpg \vbox {\unvbox\@cclv}%
301 \else

```

Note that `\boxmaxdepth` should not be set here since we wish to record the natural depth of the `holdpg` box.

This is changed so as to not lose anything, such as writes and marks, which may get into box 255 and should be returned to the list. This should only happen when the first penalty in the mechanism is discarded and therefore `\@holdpg` should always be void in this case. This can happen because a penalty is discarded whenever there is no box on the list.

It was just: `\setbox\@tempboxa \box \@cclv`.

The last box which is removed is the box put there by the double-penalty mechanism. The `\unskip` then removes the `\topskip` which is put there since the box is the first on the page.

```

302 \global \setbox\@holdpg \vbox{%
303 \unvbox\@holdpg
304 \unvbox\@cclv

```

We must now remove the box added by the float mechanism and the `\topskip` glue therefore added above it by TeX.

```

305 \setbox\@tempboxa \lastbox
306 \unskip
307 }%

```

These two are needed as separate dimensions only by `\@addmarginpar`; for other purposes we put the whole size into `\@pageht` (see below).

```

308 \@pagedp \dp\@holdpg
309 \@pageht \ht\@holdpg
310 \unvbox \@holdpg
311 \next\currbox\@currlist{%
312 \ifnum \count\currbox>\z@

```

Putting the whole size into `\@pageht` (see above).

```

313 \advance \@pageht \@pagedp
314 \ifvoid\footins \else
315 \advance \@pageht \ht\footins
316 \advance \@pageht \skip\footins
317 \advance \@pageht \dp\footins
318 \fi
319 \ifvbox \@kludgeins

```

We want to make the adjustment due to this insert only if the non-star form is used. The \*-form will probably not work with floats, but maybe it still could make some adjustment here even so?

```

320 \ifdim \wd\@kludgeins=\z@
321 \advance \@pageht \ht\@kludgeins
322 (*trace)
323 \f@trace {Extra size added: \the \ht\@kludgeins}%
324 (/trace)
325 \fi
326 \fi

```

This version puts the inserts back just before the additional material; it could be moved earlier, before unboxing the page-so-far. Neither is guaranteed not to put things on the wrong page. This version is similar to the original version.

```

327 \@reinserts
328 \@addtocurcol
329 \else
330 \@reinserts
331 \@addmarginpar
332 \fi
333 }\@latexbug

```

A 2e change: use `\addpenalty` instead of `\penalty` here. Some penalty is needed to create a potential break-point immediately after the reinserts (or the marginal). Otherwise there can be no possibility to break here and this can cause the reinserts or the marginal to appear on the next page (which is often incorrect). However, if the nobreak flag is true, a `\nobreak` must be correct.

```

334 \ifnum \outputpenalty<\z@
335 \if@nobreak
336 \nobreak
337 \else
338 \addpenalty \interlinepenalty
339 \fi
340 \fi
341 \fi
342 \fi
343 }
344 </2ekernel | fltrace>

```

(End definition for `\output` and `\@specialoutput`.)

`\@testwrongwidth` Test if the float box has the wrong width when trying to place it into some area. (Actually the test is for a conventional depth setting rather than for the width of the float. For that reason the box depth was explicitly tailored when the float was created).

```

345 <latexrelease>\IncludeInRelease{2015/01/01}%
346 <latexrelease> {\@testwrongwidth}{float order in 2-column}%
347 <*2ekernel | latexrelease | fltrace>

348 \def\@testwrongwidth #1{%
349 \ifdim\dp#1=\f@depth
350 <*trace>
351 \f1@trace{\string#1
352 \ifdim\f@depth=\z@ single \else double \fi
353 column float -- ok}%
354 </trace>
355 \else
356 \global\@testtrue
357 <*trace>
358 \f1@trace{\string#1
359 \ifdim\f@depth=\z@ double \else single \fi
360 column float -- wrong}%
361 </trace>
362 \fi}%

```

Normally looking for single column floats, which have zero depth.  
`\let\f@depth\z@`

```

364 {/2ekernel | latexrelease | fltrace}
365 \end{IncludeInRelease}
366 \IncludeInRelease{0000/00/00}%
367 \let\@testwrongwidth\@undefined
368 \let\f@depth\@undefined
369 \end{IncludeInRelease}
370

```

(End definition for `\@testwrongwidth` and `\f@depth`.)

### \@doclearpage

This is a very much an emergency action, just dumping everything: footnotes first then floats. A more sophisticated version is needed; but even more urgent is a bug-free version (see, for example, pr/3528).

Also, it puts any left-over non-boxes (writes, specials, etc.) back after any float pages created: this is a very bad bug since, for example, a kludge insert will be in quite the wrong place and, worse, be irremovable and uncancelable.

All the remaining changes are replacing the double column defer list or inserting the extra test `\@testwrongwidth{\box}` at suitable places. That is at places where a box is taken off the deferlist.

```

371 \IncludeInRelease{2015/01/01}{\@doclearpage}%
372 \def \@doclearpage {%
373 {*2ekernel | latexrelease}
374 \ifvoid\footins
375 \ifvbox\@kludgeins
376 \setbox \tempboxa \box \@kludgeins%
377 \else
378 \f@trace {kludgeins box made void}%
379 \fi
380 \else
381 \ifvbox\@tempboxa\vsplit\@cclv to\z@\unvbox\@tempboxa
382 \setbox\@tempboxa\box\@cclv
383 \xdef\@deferlist{\@toplist\@botlist\@deferlist}%
384 \global \let \@toplist \empty
385 \global \let \@botlist \empty
386 \global \colroom \colht
387 \ifx \currlist\empty
388 \else
389 \@latex@error{Float(s) lost}\ehb
390 \global \let \currlist \empty
391 \fi
392 \else
393 \makefcolumn\@deferlist
394 \whilesw\if@fcollmade \fi{\opcol\makefcolumn\@deferlist}%
395 \if@twocolumn
396 \if@firstcolumn
397 \xdef\@deferlist{\@dbltoplist\@deferlist}%
398 \global \let \dbltoplist \empty
399 \global \colht \textheight
400 \begingroup
401 \dblfloplacement

```

```

402 \@makefcolumn\@deferlist
403 \@whilesw\if@fcolmade \fi{\@outputpage
404 \@makefcolumn\@deferlist}%
405
406 \endgroup
407 \else
408 \vbox{}\clearpage
409 \fi
410 \fi

```

the next line is needed to avoid losing floats in certain circumstances a single call to the original \doclearpage will now no longer output all floats.

```

411 \ifx\@deferlist\@empty \else\clearpage \fi
412 \else
413 \setbox\@cclv\vbox{\box\@cclv\vfil}%
414 \@makecol\@opcol
415 \clearpage
416 \fi
417 }%
418 </2ekernel | latexrelease>
419 <latexrelease>\EndIncludeInRelease
420 <latexrelease>\IncludeInRelease{0000/00/00}{\@doclearpage}%
421 <latexrelease> {float order in 2-column}%
422 <latexrelease>\def \@doclearpage {%
423 \ifvoid\footins

```

We empty any left over kludge insert box here; this is a temporary fix. It should perhaps be applied to one page of cleared floats, but who cares? The whole of this stuff needs completely redoing for many such reasons.

```

424 <latexrelease> \ifvbox\@kludgeins
425 <latexrelease> {\setbox \@tempboxa \box \@kludgeins}%
426 <*trace>
427 <latexrelease> \f1@trace {\kludgeins box made void}%
428 </trace>
429 <latexrelease> \fi
430 <latexrelease> \setbox\@tempboxa\vsplit\@cclv to\z@\unvbox\@tempboxa
431 <latexrelease> \setbox\@tempboxa\box\@cclv
432 <latexrelease> \xdef\@deferlist{\@toplist\@botlist\@deferlist}%
433
434 <latexrelease> \global \let \@toplist \@empty
435 <latexrelease> \global \let \@botlist \@empty
436 <latexrelease> \global \@colroom \@colht
437 <latexrelease> \ifx \@currlist\@empty
438 <latexrelease> \else
439 <latexrelease> \@latexerr{Float(s) lost}\@ehb
440 <latexrelease> \global \let \@currlist \@empty
441 <latexrelease> \fi
442 <latexrelease> \@makefcolumn\@deferlist
443 <latexrelease> \@whilesw\if@fcolmade \fi
444 <latexrelease> {\@opcol\@makefcolumn\@deferlist}%
445 <latexrelease> \if@twocolumn
446 <latexrelease> \if@firstcolumn
447 <latexrelease> \xdef\@dbldeferlist{\@dbltoplist\@dbldeferlist}%

```

```

446 <|latexrelease> \global \let \@dbltoplist \empty
447 <|latexrelease> \global \@colht \textheight
448 <|latexrelease> \begingroup
449 <|latexrelease> \@dblfloatingplacement
450 <|latexrelease> \@makefcolumn\@dbldeflist
451 <|latexrelease> \@whilesw\if@fcolmade \fi
452 <|latexrelease> {\@outputpage\@makefcolumn\@dbldeflist}%
453 <|latexrelease> \endgroup
454 <|latexrelease> \else
455 <|latexrelease> \vbox{}\clearpage
456 <|latexrelease> \fi
457 <|latexrelease> \fi
458 <|latexrelease> \else
459 <|latexrelease> \setbox\@cclv\vbox{\box\@cclv\vfil}%
460 <|latexrelease> \@makecol\@opcol
461 <|latexrelease> \clearpage
462 <|latexrelease> \fi
463 <|latexrelease> }%
464 <|latexrelease>\EndIncludeInRelease

```

(End definition for \@doclearpage.)

\@opcol Several changes in detail here.

```

465 {*2ekernel | fltrace}
466 \def \@opcol {%
467 \if@twocolumn
468 \@outputdblcol
469 \else
470 \@outputpage
471 {*trace}
472 \fl@trace{PAGE: one column (float? see above) page completed}%
473 }
```

Not needed since it comes after \@outputpage:

```

474 % \global\@colht\textheight
475 \fi
```

These do not need to be done every time \@opcol is used: they should be grouped together since they all need to be done at the end of the non-special output routine, or at the end of a clearpage one.

```

476 \global \z@\global \z@ \global \textfloatsheight \z@
477 \@floatplacement
478 }
479
```

(End definition for \@opcol.)

\@makecol We must rewrite this macro to allow for variations in page-makeup required by changes in page-length.

This uses a different macro if a special-length column is being produced.

```

480 {*2ekernel}
481 \gdef \@makecol {%
482 \ifvoid\footins
483 \setbox\@outputbox \box\@cclv
484 \else
485 \setbox\@outputbox \vbox {%
```

This `\boxmaxdepth` setting is to ensure that deep footnotes do not overwrite the footer (on account of the negative skip added later): it should use `\@maxdepth` otherwise the change is pointless when there are footnotes.

But see also its use when combining floats.

```

486 \boxmaxdepth \@maxdepth
487 % \@tempdima\dp\@cclv
488 % \unvbox \@cclv
489 % \vskip-\@tempdima
490 \vskip \skip\footins
491 \color@begingroup
492 \normalcolor
493 \footnoterule
494 \unvbox \footins
495 \color@endgroup
496 }%
497 \fi

```

The h floats have now been finally committed to this page so we can reset their list. The top and bottom floats are then added to the page.

```

498 \let\@elt\relax
499 \xdef\@freelist{\@freelist\@midlist}%
500
501 \global \let \@midlist \empty
502 \@combinefloats

```

The variations start here in case `\enlargethispage` has been used.

```

502 \ifvbox\@kludgeins
503 \@makespecialcolbox
504 \else

```

This extra reboxing is only needed to add the `\@texttop` and `\@textbottom` but this could be done earlier, when the floats are added.

The `\boxmaxdepth` resetting here will have no effect unless `\@textbottom` ends with a box or rule. So is this (or possibly `\@maxdepth`) the correct value?

The `\vskip -\dimen0` ensures that the visible depth of the box does not affect the placement of anything on the page. Thus very deep pages will overprint the footer; but these should have been prevented by suitable settings of the maxdepths at appropriate times.

If `\@textbottom` ends with a box or rule of non-zero depth then this skip adjustment should be done again after it.

I think that the final boxing of the main text page could have a common ending which may make it simpler to see what is going on.

This needs further investigation, especially in the ‘special case’.

Also, the `\boxmaxdepth` setting here affects what happens within `\@texttop` and `\@textbottom`, should it? Is it needed at all?

RmS 91/10/22: Replaced `\dimen128` by `\dimen0`.

```

505 \setbox\@outputbox \vbox to\@colht {%
506 % \boxmaxdepth \maxdepth %??
507 \@texttop
508 \dimen0 \dp\@outputbox
509 \unvbox \@outputbox

```

```

510 \vskip -\dimen@%
511 \textbottom
512 }%
513 \fi
514 \global \maxdepth \maxdepth
515 }

```

(End definition for `\@makecol`.)

- `\@reinserts` This is the code which reinserts the inserts. It puts them all in one place; this can make some of them come out on the wrong page. It has been put into a separate macro to expedite experimentation.

```

516 \gdef \@reinserts{%
517 \ifvoid\footins\else\insert\footins{\unvbox\footins}\fi
518 \ifvbox\@kludgeins\insert\@kludgeins
519 {\unvbox\@kludgeins}\fi
520 }
521 </2ekernel>

```

(End definition for `\@reinserts`.)

- `\@makespecialcolbox` This implements certain variations in page-makeup.

```

522 <*2ekernel | ftrace>
523 \gdef \@makespecialcolbox {%
524 <*trace>
525 \ftrace{Kludgeins ht \the\ht\@kludgeins\space
526 dp \the\dp\@kludgeins\space
527 wd \the\wd\@kludgeins}%
528 </trace>

```

First we find the natural height of the column.

See above for discussion of what is happening here.

This needs further investigation, especially in this ‘special case’.

```

529 \setbox\@outputbox \vbox {%
530 \texttop
531 \dimen@ \dp\@outputbox
532 \unvbox\@outputbox
533 \vskip-\dimen@
534 }%
535 \tempdima \colht
536 \ifdim \wd\@kludgeins>\z@

```

Note that in this case (the `*`-version), the height of the `\@kludgeins` box is not used since its value is somewhat arbitrary: it need only be big enough to ensure that the page-break is not taken prematurely.

Here we calculate how much vertical space needs to be added in order to enable the column to fit into a box of size `\@colht` using the best information we have about the amount of shrink available (another thing which is known internally about a box, but cannot be accessed at the TeX level!).

This needs TeX3 otherwise `\pageshrink` is zero anyway; it may not be exactly the figure we wish as it is the total available from the all the material collected before the page-break decision is made. It will, we think, always be an overestimate of the actual shrink in the box; therefore this should always force the shortest possible column with the possibility of an overfull box.

This should work for both flush- and ragged-bottom setting since it makes the contents no smaller than the size (`\@colht`) of the box into which they are put.

Their should perhaps be an upper limit, of `0pt?`, on the extra space added to force shrinking.

See above for a discussion of the `\boxmaxdepth` setting here.

```

537 \advance \@tempdima -\ht\@outputbox
538 \advance \@tempdima \pageshrink
539 {*trace}
540 \f@trace {Natural ht of col: \the \ht\@outputbox}%
541 \f@trace {\string \@colht: \the \@colht}%
542 \f@trace {Pageshrink added: \the \pageshrink}%
543 \f@trace {Hence, space added: \the \@tempdima}%
544 {/trace}
545 \setbox\@outputbox \vbox to \@colht {%
546 %
547 \boxmaxdepth \maxdepth
548 \unvbox\@outputbox
549 \vskip \@tempdima
550 \textbottom
551 }%

```

For the unstarred version, the final size of the page is precisely specified. Therefore, at least for the flush-bottom case, we need to ensure that, visually, it has this size exactly.

Thus we calculate this size and set the material in a box of this size, which is then put into a box of size `\@colht` with `\vss` at the bottom.

```

551 \else
552 \advance \@tempdima -\ht\@kludgeins
553 {*trace}
554 \f@trace {Natural ht of col: \the \ht\@outputbox}%
555 \f@trace {\string \@colht: \the \@colht}%
556 \f@trace {Extra size added: -\the \ht \@kludgeins}%
557 \f@trace {Hence, height of inner box: \the \@tempdima}%
558 \f@trace {Max? pageshrink available: \the \pageshrink}%
559 {/trace}

```

This type of final packaging could be done always; this may simplify all of this page-makeup.

It is not necessary to set `\boxmaxdepth` here since the `\@outputbox` ends with glue.

```

560 \setbox \@outputbox \vbox to \@colht {%
561 \vbox to \@tempdima {%
562 \unvbox\@outputbox
563 \textbottom}%
564 \vss}%
565 \fi

```

Finally we need to explicitly make the insert box void.

```

566 {\setbox \@tempboxa \box \@kludgeins}%
567 {*trace}
568 \f@trace {kludgeins box made void}%
569 {/trace}
570 }
571 {/2ekernel | ftrace}

```

(End definition for `\makespecialcolbox`.)

\@texttop These do nothing as a default.

```

\@textbottom
 572 {*2ekernel}
 573 \let \@texttop \relax
 574 \let \@textbottom \relax

```

(End definition for \@texttop and \@textbottom.)

\@resetactivechars RmS 93/09/06: added hook to protect against certain active characters in the output routine. Default checks are for active space and end-of-line.

```

 575 \def\@activechar@info #1{%
 576 \@latex@info@no@line {Active #1 character found while
 577 output routine is active
 578 \MessageBreak
 579 This may be a bug in a package file
 580 you are using}%
 581 }

```

Do not put any spaces in this next bit!

```

 582 \begingroup
 583 \obeylines\obeyspaces%
 584 \catcode`\'\active%
 585 \gdef\@resetactivechars{%
 586 \def^~M{\@activechar@info{EOL}\space}%
 587 \def {\@activechar@info{space}\space}%
 588 \let'\active@math@prime}%
 589 \endgroup

```

(End definition for \@resetactivechars and \@activechar@info.)

\@outputpage \@shipoutsetup \@writesetup The \color@hbox hooks here are used to avoid putting just a colour special into an otherwise empty box (in a header or footer). These boxes are often set to be completely empty and so adding a special produces a very underfull box message.

There has been extensive tidying up of the old code here; including the removal of a level of grouping.

The setting of \protect immediately before the \shipout is needed so that protected commands within \writes are handled correctly.

Within shipout's vbox it is reset to its default value, \relax.

Resetting it to its default value after the shipout has been completed (and the contents of the writes have been expanded) must be done by use of \aftergroup. This is because it must have the value \relax before macros coming from other uses of \aftergroup within this box are expanded.

Putting this into the \aftergroup token list does not affect the definition used in expanding the \writes because the aftergroup token list is only constructed when popping the save-stack, it is not expanded until after the shipout is completed.

Question: should things from an \aftergroup within the shipped out box be executed in the environment set up for the writes, or after it finishes?

A lot of this code has been in-lined to prevent mis-use of internal commands as hooks.

```

 590 {/2ekernel}
 591 {latexrelease}\IncludeInRelease{2017/04/15}%
 592 {latexrelease} {\@outputpage}{Reset language for hyphenation}%
 593 {*2ekernel | latexrelease}
 594 \def\@outputpage{%

```

The `\endgroup` is put in by `\aftergroup`.

595   `\begingroup`

Now all the set-up stuff has been in-lined for Frank.

First the stuff for the writes.

From here ... was in the command `\@writesetup`.

596   `\let \protect \noexpand`

RmS 93/08/19: Redefined accents to allow changes in font encoding; but exactly why was this needed?

Reset `\language` to the value current at `\begin{document}`. In particular this ensures that a pagebreak in `\verb+im` does not prevent hyphenation in the page head.

597   `\language\document@default@\language`

The `\catcode`\\ = 10` was removed as it was considered useless (presumably because nothing gets tokenized during shipout).

This was put in as some error produced active spaces in a mark, I think.

Why was the hyphen reset?

598   `\@resetactivechars`

If a page break happens between the start of a list and its first item the `@newlist` will be true and this will mess up any list that is used in the header or footer of the page. So we have to reset that flag.

599   `\global\let\@if@newlist\if@newlist`

600   `\global@\newlistfalse`

This next hook replaces the following:

```
\let\-\@dischyp
\let'\@acci\let`\@acci\let\=\@acciii
\let\\@\normalcr
\let\par\@@par %% 15 Sep 87 (this was once inside the box)
```

and it does more than they did; in particular it sets:

```
\parindent\z@
\parskip\z@skip
\everypar{}%
\leftskip\z@skip
\rightskip\z@skip
\parfillskip\@flushglue
\lineskip\normallineskip
\baselineskip\normalbaselineskip
\sloppy

601 \@parboxrestore
```

... to here was in the command `\@writesetup`.

602   `\shipout \vbox{%`

603    `\set@typeset@protect`

604    `\aftergroup \endgroup`

Correct? or just restore by ending the group?

605   `\aftergroup \set@typeset@protect`

This first bit has been moved inside the shipped out box.

Now the setup inside the shipped out box; this should contain all the stuff that could only affect typesetting; other stuff may need to be reset for the writes also.

From here ... was in the command \cshipoutsetup.

```
606 \if@specialpage
607 \global\@specialpagefalse\@nameuse{ps@\@specialstyle}%
608 \fi
609 \if@twoside
610 \ifodd\count\z@ \let\@thehead\@oddhead \let\@thefoot\@oddfoot
611 \let\@themargin\oddsidemargin
612 \else \let\@thehead\@evenhead
613 \let\@thefoot\@evenfoot \let\@themargin\evensidemargin
614 \fi
615 \fi
```

The rest was always inside the box.

RmS 91/08/15: added this line:

```
616 \reset@font
```

RmS 93/08/06 Added \lineskiplimit=0pt to guard against it being nonzero: e.g. by \offinterlineskip being in effect.

There are probably lots of other things that may need resetting.

```
617 \normalsize
```

Reset the space factors.

```
618 \normalsfcodes
```

Reset these here (previously reset separately for head and foot)

```
619 \let\label\@gobble
620 \let\index\@gobble
621 \let\glossary\@gobble
622 \baselineskip\z@skip \lineskip\z@skip \lineskiplimit\z@
```

... to here was in the command \cshipoutsetup.

```
623 \begindvi
624 \vskip \topmargin
625 \moveoveright\@themargin \vbox {%
626 \setbox\@tempboxa \vbox to\headheight{%
627 \vfil
628 \color@hbox
629 \normalcolor
630 \hb@xt@\textwidth{\@thehead}%
631 \color@endbox
632 }%
633 \dp\@tempboxa \z@
634 \box\@tempboxa
635 \vskip \headsep
636 \box\@outputbox
637 \baselineskip \footskip
638 \color@hbox
639 \normalcolor
640 \hb@xt@\textwidth{\@thefoot}%
641 \color@endbox
642 }
```

22 Feb 87

```
632
633 \dp\@tempboxa \z@
634 \box\@tempboxa
635 \vskip \headsep
636 \box\@outputbox
637 \baselineskip \footskip
638 \color@hbox
639 \normalcolor
640 \hb@xt@\textwidth{\@thefoot}%
641 \color@endbox
```

```

642 }%
643 }%
\endgroup now inserted by \aftergroup
 Restore \if@newlist
644 \global\let\if@newlist\@@if@newlist
645 \global \colht \textheight
646 \stepcounter{page}%

```

It is now clear that this does something useful, thanks to Piet van Oostrum. It is needed because a float page is made without using TeX's page-builder; thus the output routine is never called so the marks are not updated.

```

647 \let\firstmark\botmark
648 }
649 </2ekernel | latexrelease>
650 <latexrelease>\EndIncludeInRelease
651 <latexrelease>\IncludeInRelease{0000/00/00}%
652 <latexrelease> {\@outputpage}{Reset language for hyphenation}%
653 <latexrelease>\def\@outputpage{%
654 <latexrelease>\begingroup
655 <latexrelease> \let \protect \noexpand
656 <latexrelease> \resetactivechars
657 <latexrelease> \global\let\@@if@newlist\if@newlist
658 <latexrelease> \global\@newlistfalse
659 <latexrelease> \parboxrestore
660 <latexrelease> \shipout \vbox{%
661 <latexrelease> \set@typeset@protect
662 <latexrelease> \aftergroup \endgroup
663 <latexrelease> \aftergroup \set@typeset@protect
664 <latexrelease> \if@specialpage
665 <latexrelease> \global\@specialpagefalse\@nameuse{ps@\@specialstyle}%
666 <latexrelease> \fi
667 <latexrelease> \if@twoside
668 <latexrelease> \ifodd\count\z@%
669 <latexrelease> \let\@thehead\@oddhead \let\@tfoot\@oddfoot
670 <latexrelease> \let\@themargin\oddsidemargin
671 <latexrelease> \else \let\@thehead\@evenhead
672 <latexrelease> \let\@tfoot\@evenfoot \let\@themargin\evensidemargin
673 <latexrelease> \fi
674 <latexrelease> \fi
675 <latexrelease> \reset@font
676 <latexrelease> \normalsize
677 <latexrelease> \normalsfcodes
678 <latexrelease> \let\label\@gobble
679 <latexrelease> \let\index\@gobble
680 <latexrelease> \let\glossary\@gobble
681 <latexrelease> \baselineskip\z@skip \lineskip\z@skip \lineskiplimit\z@
682 <latexrelease> \begindvi
683 <latexrelease> \vskip \topmargin
684 <latexrelease> \overright\@themargin \vbox {%
685 <latexrelease> \setbox\@tempboxa \vbox to\headheight{%
686 <latexrelease> \vfil
687 <latexrelease> \color@hbox
688 <latexrelease> \normalcolor

```

```

689 〈latexrelease〉 \hb@xt@{\textwidth}{\@thehead}%
690 〈latexrelease〉 \color@endbox
691 〈latexrelease〉 }%
692 〈latexrelease〉 \dp@\tempboxa \z@
693 〈latexrelease〉 \box@\tempboxa
694 〈latexrelease〉 \vskip \headsep
695 〈latexrelease〉 \box@\outputbox
696 〈latexrelease〉 \baselineskip \footskip
697 〈latexrelease〉 \color@hbox
698 〈latexrelease〉 \normalcolor
699 〈latexrelease〉 \hb@xt@{\textwidth}{\@thefoot}%
700 〈latexrelease〉 \color@endbox
701 〈latexrelease〉 }%
702 〈latexrelease〉 }%
703 〈latexrelease〉 \global\let\if@newlist\@if@newlist
704 〈latexrelease〉 \global \colht \textheight
705 〈latexrelease〉 \stepcounter{page}%
706 〈latexrelease〉 \let\firstmark\botmark
707 〈latexrelease〉}
708 〈latexrelease〉\EndIncludeInRelease
709 {*2ekernel}

```

(*End definition for \@outputpage, \@shipoutsetup, and \@writesetup.*)

\@begindvi This unboxes stuff that must appear before anything else in the .dvi file, then returns that box register to the free list and cancels itself.

The stuff in the box should not add any typeset material to the page.

```

710 \def \@begindvi{%
711 \unvbox \@begindvibox
712 \global\let \@begindvi \empty
713 }

```

(*End definition for \@begindvi.*)

\@combinefloats The \boxmaxdepth setting here was not made local to a box so was dangerous. It is needed only within the box made by \@cflt (and not normally even there), so it has been moved there; this also agrees with the original pseudocode.

```

714 \def \@combinefloats{%
715 % \boxmaxdepth \maxdepth
716 \ifx \@toplist\empty \else \@cflt \fi
717 \ifx \@botlist\empty \else \@cflb \fi
718 }

719 \def \@cflt{%
720 \let \@elt \@comflelt
721 \setbox\@tempboxa \vbox{}%
722 \@toplist
723 \setbox\@outputbox \vbox{%
724 \boxmaxdepth \maxdepth
725 \unvbox\@tempboxa
726 \vskip -\floatsep
727 \topfigrule
728 \vskip \textfloatsep
729 \unvbox\@outputbox

```

```

730 }%
731 \let\@elt\relax
732 \xdef\@freelist{\@freelist\@toplist}%
733 \global\let\@toplist\@empty
734 }

735 \def \@cflb {%
736 \let\@elt\@comflelt
737 \setbox\@tempboxa \vbox{}%
738 \@botlist
739 \setbox\@outputbox \vbox{%
740 \unvbox\@outputbox
741 \vskip \textfloatsep
742 \botfigrule
743 \unvbox\@tempboxa
744 \vskip -\floatsep
745 }%
746 \let\@elt\relax
747 \xdef\@freelist{\@freelist\@botlist}%
748 \global \let \@botlist\@empty
749 }

```

(End definition for `\@combinefloats`, `\@cflt`, and `\@cflb`.)

```

\@comflelt
\@comdblfllelt 750 \def\@comflelt#1{\setbox\@tempboxa
751 \vbox{\unvbox\@tempboxa\box #1\vskip\floatsep}}
752 \def\@comdblfllelt#1{\setbox\@tempboxa
753 \vbox{\unvbox\@tempboxa\box #1\vskip\dblfloatsep}}
754 \def \@combinedblfloats{%
755 \ifx \@dbltoplist \@empty
756 \else
757 \setbox\@tempboxa \vbox{}%
758 \let \@elt \@comdblfllelt
759 \@dbltoplist
760 \let \@elt \relax
761 \xdef \@freelist {\@freelist\@dbltoplist}%
762 \global\let \@dbltoplist \@empty
763 \setbox\@outputbox \vbox to\textheight

```

The setting of `\boxmaxdepth` here has no effect since the `\@outputbox` should already have depth zero. Even so, it would have no effect on the layout of the page.

```

764 {%\boxmaxdepth\maxdepth %% probably not needed, CAR
765 \unvbox\@tempboxa\vskip-\dblfloatsep

```

Here we need different typesetting if the top float comes from `\@topnewpage`.

```

766 \ifnum \@dbltopnum>\m@ne
767 \dblfigrule
768 \fi
769 \vskip \dbltextfloatsep

```

If pdf links are present in the galley and those links get broken across pages they have to end up being on the same level of boxing (even if not actually in the same structure) due to some engine restrictions in pdfTeX and LuaTeX. We therefore unbox `\@outputbox`

here (which only contains a single `\hbox`) so that this case has the same boxing level as a normal twocolumn page without top floats.

```

770 \unvbox\@outputbox
771 }%
772 \fi
773 }
774 </2ekernel>

```

(End definition for `\@comfleft`, `\@comdblleft`, and `\@combinedblfloats`.)

`\@startcolumn` `\@startdblcolumn` We could combine (most of) these two into `\@startcol <list>`. Note that `\@xstartcol` was only used once (i.e. in `\@startcolumn`); it has therefore been removed. This is not quite as efficient but it now has the same structure as `\@startdblcolumn`.

The empty-list test has been moved to `\@tryfcolumn`.

```

775 <*2ekernel | fltrace>
776 \def \@startcolumn {%
777 \global \@colroom \@colht
778 \@tryfcolumn \@deferlist
779 \if@fcolmade
780 (*trace)
781 \fl@trace{PAGE: float \if@twocolumn column \else page \fi
782 completed}%
783 }/trace>
784 \else
785 \begingroup
786 \let \reserved@b \@deferlist
787 \global \let \@deferlist \@empty
788 \let \@elt \cscoelt
789 \reserved@b
790 \endgroup
791 \fi
792 }

```

This one does not need to set `\@colht`.

```

793 </2ekernel | fltrace>
794 <|latexrelease | fltrace>\IncludeInRelease{2015/01/01}%
795 <|latexrelease | fltrace> {\@startdblcolumn}{float order in 2-column}%
796 <*2ekernel | latexrelease | fltrace>
797 \def \@startdblcolumn {%
798 \@tryfcolumn \@deferlist
799 \if@fcolmade
800 <fltrace> \fl@trace{PAGE: double float page completed}%
801 \else
802 \begingroup
803 \let \reserved@b \@deferlist
804 \global \let \@deferlist \@empty
805 \let \@elt \csdblcoelt
806 \reserved@b
807 \endgroup
808 \fi
809 }
810 </2ekernel | latexrelease | fltrace>
811 <|latexrelease | fltrace>\EndIncludeInRelease

```

```

812 〈latexrelease | fltrace〉\IncludeInRelease{0000/00/00}%
813 〈latexrelease | fltrace〉 {\@startdblcolumn}{float order in 2-column}%
814 〈latexrelease | fltrace〉\def \@startdblcolumn {%

```

Not needed since this always comes after \@outputpage:

```

815 〈latexrelease | fltrace〉% \global \colht \textheight
816 〈latexrelease | fltrace〉 \@tryfc \dbldeferlist
817 〈latexrelease | fltrace〉 \if@fcolmade
818 〈*trace〉
819 〈latexrelease | fltrace〉 \fl@trace{PAGE: double float page completed}%
820 〈/trace〉
821 〈latexrelease | fltrace〉 \else
822 〈latexrelease | fltrace〉 \begingroup
823 〈latexrelease | fltrace〉 \let \reserved@b \dbldeferlist
824 〈latexrelease | fltrace〉 \global \let \dbldeferlist \empty
825 〈latexrelease | fltrace〉 \let \elt \sdblcolelt
826 〈latexrelease | fltrace〉 \reserved@b
827 〈latexrelease | fltrace〉 \endgroup
828 〈latexrelease | fltrace〉 \fi
829 〈latexrelease | fltrace〉}%
830 〈latexrelease | fltrace〉\EndIncludeInRelease
831 〈*2ekernel | fltrace〉

```

(End definition for \@startcolumn and \@startdblcolumn.)

\@tryfc Now tests if its list is empty before any further exertion.

```

832 \def \@tryfc #1{%
833 \global \fcolmadefalse
834 \ifx #1\empty
835 \else
836 〈*trace〉
837 \fl@trace{PAGE: try float \if@twocolumn column/page\else page\fi
838 ---\string #1}%
839 \fl@trace{---- \string #1: #1}%
840 〈/trace〉
841 \xdef\@trylist{#1}%
842 \global \let \failedlist \empty
843 \begingroup
844 \let \elt \tryfc \@trylist
845 \endgroup
846 \if@fcolmade
847 \vtryfc #1%
848 \fi
849 \fi
850 }
851 〈/2ekernel | fltrace〉

```

(End definition for \@tryfc.)

```
852 〈*2ekernel〉
```

\@scolelt

```
853 \def \@scolelt#1{\def \@currbox{#1}\addtonextcol}
```

(End definition for \@scolelt.)

```
\@sdblcolelt
854 \def\@sdblcolelt#1{\def\@currbox{#1}\@addtoblcol}
(End definition for \@sdblcolelt.)
```

```
\@vtryfc
855 \def\@vtryfc #1{%
856 \global\setbox\@outputbox\vbox{}%
857 \let\@elt\@wtryfc
858 \@flsucceed
859 \global\setbox\@outputbox \vbox to\@colht{%
860 \vskip \@fptop
861 \vskip -\@fpsep
862 \unvbox \@outputbox
863 \vskip \@fpbot}%
864 \let\@elt\relax
865 \xdef #1{\@failedlist\@flfail}%
866 \xdef\@freelist{\@freelist\@flsucceed}}
```

(End definition for \@vtryfc.)

```
\@wtryfc
867 \def\@wtryfc #1{%
868 \global\setbox\@outputbox\vbox{%
869 \unvbox\@outputbox
870 \vskip\@fpsep
871 \box #1}}
```

(End definition for \@wtryfc.)

```
\@xtryfc
872 </2ekernel>
873 <|latexrelease|\IncludeInRelease{2015/01/01}{\@xtryfc}%
874 <|latexrelease> {float order in 2-column}%
875 <*2ekernel | latexrelease>
876 \def\@xtryfc #1{%
877 \next\reserved@a\@trylist{}{}%
878 \currtype \count #1%
879 \divide\currtype\@xxxii
880 \multiply\currtype\@xxxii
881 \bitor \currtype \@failedlist
882 \testfp #1%
883 \testwidth #1%
884 \ifdim \ht #1>\@colht
885 \testtrue
886 \fi
887 \if@test
888 \cons\@failedlist #1%
889 \else
890 \xtryfc #1%
891 \fi}%
892 </2ekernel | latexrelease>
```

```

893 〈\latexrelease〉\EndIncludeInRelease
894 〈\latexrelease〉\IncludeInRelease{0000/00/00}{\@xtryfc}%
895 〈\latexrelease〉 {float order in 2-column}%
896 〈\latexrelease〉\def\@xtryfc #1{%
897 〈\latexrelease〉 \cnext\reserved@a\@trylist{}{}%
898 〈\latexrelease〉 \currtype \count #1%
899 〈\latexrelease〉 \divide\currtype\@xxxii
900 〈\latexrelease〉 \multiply\currtype\@xxxii
901 〈\latexrelease〉 \bitor \currtype \@failedlist
902 〈\latexrelease〉 \@testfp #1%
903 〈\latexrelease〉 \ifdim \ht #1>\@colht
904 〈\latexrelease〉 \@testtrue
905 〈\latexrelease〉 \fi
906 〈\latexrelease〉 \if@test
907 〈\latexrelease〉 \cons\@failedlist #1%
908 〈\latexrelease〉 \else
909 〈\latexrelease〉 \@ytryfc #1%
910 〈\latexrelease〉 \fi}%
911 〈\latexrelease〉\EndIncludeInRelease
912 〈*2ekernel〉

```

(End definition for \@xtryfc.)

```
\@ytryfc
913 \def\@ytryfc #1{%
914 \begingroup
915 \gdef\@flsucceed{\@elt #1}%
916 \global\let\@flfail\empty
917 \tempdima\ht #1%
918 \let\@elt\@ztryfc
919 \@trylist
920 \ifdim \tempdima >\@fpmin
921 \global\fcolmadetrue
922 \else
923 \cons\@failedlist #1%
924 \fi
925 \endgroup
926 \if\fcolmade
927 \let\@elt\@gobble
928 \fi}

```

(End definition for \@ytryfc.)

```
\@ztryfc
929 〈/2ekernel〉
930 〈\latexrelease〉\IncludeInRelease{2015/01/01}{\@ztryfc}%
931 〈\latexrelease〉 {float order in 2-column}%
932 〈*2ekernel | \latexrelease〉
933 \def\@ztryfc #1{%
934 \tempcnta\count #1%
935 \divide\tempcnta\@xxxii
936 \multiply\tempcnta\@xxxii
937 \bitor \tempcnta {\@failedlist \@flfail}%
938 \testfp #1%

```

```

not in fixfloats?

939 \@testwrongwidth #1%
940 \@tempdimb\@tempdima
941 \advance\@tempdimb\ht #1%
942 \advance\@tempdimb\@fpsep
943 \ifdim \@tempdimb >\@colht
944 \@testtrue
945 \fi
946 \if@test
947 \@cons\@flfail #1%
948 \else
949 \@cons\@flsucceed #1%
950 \@tempdima\@tempdimb
951 \fi}%
952 {/2ekernel | latexrelease}
953 {/latexrelease}\EndIncludeInRelease
954 {/latexrelease}\IncludeInRelease{0000/00/00}{\ztryfc}%
955 {/latexrelease} {float order in 2-column}%
956 {/latexrelease}\def\@ztryfc #1{%
957 {/latexrelease} \@tempcnta \count#1%
958 {/latexrelease} \divide\@tempcnta\@xxxii
959 {/latexrelease} \multiply\@tempcnta\@xxxii
960 {/latexrelease} \or\@tempcnta {\@failedlist \@flfail}%
961 {/latexrelease} \@testfp #1%
962 {/latexrelease} \@tempdimb\@tempdima
963 {/latexrelease} \advance\@tempdimb \ht#1%
964 {/latexrelease} \advance\@tempdimb\@fpsep
965 {/latexrelease} \ifdim \@tempdimb >\@colht
966 {/latexrelease} \@testtrue
967 {/latexrelease} \fi
968 {/latexrelease} \if@test
969 {/latexrelease} \@cons\@flfail #1%
970 {/latexrelease} \else
971 {/latexrelease} \@cons\@flsucceed #1%
972 {/latexrelease} \@tempdima\@tempdimb
973 {/latexrelease} \fi}%
974 {/latexrelease}\EndIncludeInRelease

```

(End definition for \ztryfc.)

The major changes for float suppression and the changes to the float mechanism to make it conform to the documentation are in these next macros.

\@addtobot Lots of changes.

```

975 {/*2ekernel | ftrace}
976 \def \@addtobot {%
977 {*trace}
978 \f@trace{***Start addtobot}%
979 {/trace}
980 \getfpsbit 4\relax
981 {*trace}
982 \f@trace{fpstype \ifodd \@tempcnta OK \else not \fi bot:
983 \the \@fpstype}%
984 {/trace}

```

```

985 \ifodd \@tempcnta
986 \@flsetnum \@botnum
987 \ifnum \@botnum>\z@
988 \atempswafalse
989 \@flcheckspace \@botroom \@botlist
990 \if@tempswa

```

This next line means that this page is produced with box 255 having depth zero, rather than the normal maxdepth: is this needed, useful?

```

991 \global \maxdepth \z@
992 \@flupdates \@botnum \@botroom \@botlist
993 (*trace)
994 \@fl@trace{colroom (after-bot) = \the \@colroom}%
995 \@fl@trace{colnum (after-bot) = \the \@colnum}%
996 \@fl@trace{botnum (after-bot) = \the \@botnum}%
997 \@fl@trace{***Success: bot}%
998 (/trace)
999 \@inserttrue
1000 \fi
1001 (*trace)
1002 \else
1003 \@fl@trace{Fail: botnum = \the \@botnum:
1004 fpstype \the \@fpstype=ORD?}%
1005 \ifnum \@fpstype<\sixt@n
1006 \@fl@trace{ERROR: !b float not successful (addtobot)}%
1007 \fi
1008 (/trace)
1009 \fi
1010 \fi
1011 }

```

(End definition for \@addtobot.)

\@addtotoporbot Lots of changes.

```

1012 \def \@addtotoporbot {%
1013 (*trace)
1014 \@fl@trace{***Start addtotoporbot}%
1015 (/trace)
1016 \@getfpsbit \tw@
1017 (*trace)
1018 \@fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi top:
1019 \the \@fpstype}%
1020 (/trace)
1021 \ifodd \@tempcnta
1022 \@flsetnum \@topnum
1023 \ifnum \@topnum>\z@
1024 \atempswafalse
1025 \@flcheckspace \@toproom \@topl
1026 \if@tempswa
1027 \obitor\currtype{\@midlist\@botlist}%
1028 (*trace)
1029 \@fl@trace{(mid+bot)list: \@midlist, \@botlist:
1030 (addtotoporbot-before)}%
1031 (/trace)

```

```

1032 \if@test
1033 <*trace>
1034 \f@trace{type already on list: mid or bot---sent to addtobot}%
1035 </trace>
1036 \else
1037 \f@lupdates \@topnum \@toproom \@topl
1038 <*trace>
1039 \f@trace{colroom (after-top) = \the \@colroom}%
1040 \f@trace{colnum (after-top) = \the \@colnum}%
1041 \f@trace{topnum (after-top) = \the \@topnum}%
1042 \f@trace{***Success: top}%
1043 </trace>
1044 \@inserttrue
1045 \fi
1046 \fi
1047 <*trace>
1048 \else
1049 \f@trace{Fail: topnum = \the \@topnum: fpstype
1050 \the \@fpstype=ORD?}%
1051 \ifnum \@fpstype<\sixt@n
1052 \f@trace{ERROR: !t float not successful (addtotoporbot)}%
1053 \fi
1054 </trace>
1055 \fi
1056 \fi
1057 \if@insert
1058 \else
1059 <*trace>
1060 \f@trace{sent to addtobot (addtotoporbot)}%
1061 </trace>
1062 \@addtobot
1063 \fi
1064 }
1065 </2ekernel | fltrace>

(End definition for \@addtotoporbot.)

```

\@addtocurcol Lots of changes.

```

1066 <latxrelease | fltrace | flafter> \IncludeInRelease{2015/01/01}%
1067 <latxrelease | fltrace | flafter> {\@addtocurcol}{float order in 2-column}%
1068 <2ekernel | latxrelease | fltrace | flafter>
1069 \def \@addtocurcol {%
1070 <*trace>
1071 \f@trace{***Start addtocurcol}%
1072 </trace>
1073 \@insertfalse
1074 \@setfloattypecounts
1075 \ifnum \@fpstype=8
1076 <*trace>
1077 \f@trace{fpstype !p only (addtocurcol): \the \@fpstype = 8?}%
1078 </trace>
1079 \else
1080 \ifnum \@fpstype=24
1081 <*trace>

```

```

1082 \fl@trace{fpstype p only (addtocurcol): \the \fpstype = 24?}%
1083
```

```
</trace>
```

```
\else
```

```
\@fsettextmin
```

This is a new adjustment which is quite a major change in functionality; but it implements the documentation. Note that \reqcolroom will include the whole of the page-so-far, and hence includes \textfloatsheight of floats, so before comparing it with \textmin, we add this to \textmin also.

```

1086
```

```
<*trace>
```

```
\fl@trace{textfloatsheight (before) = \the \textfloatsheight}%
```

```
</trace>
```

```
\advance \textmin \textfloatsheight
```

```
1089
```

```
\reqcolroom \pageht
```

```
1090
```

```
\@pageht
```

```
This line must be removed since \specialoutput changed.
```

```

1091 % \advance \reqcolroom \pageht
1092
```

```
<*trace>
```

```
\fl@trace{textmin + textfloatsheight: \the \textmin}%

```

```
1094
```

```
\fl@trace{page-so-far: \the \reqcolroom}%

```

```
</trace>
```

```
\ifdim \textmin > \reqcolroom
```

```
\reqcolroom \textmin
```

```
<*trace>
```

```
\fl@trace{ORD? textmin being used}%

```

```
</trace>
```

```
\fi
```

```
\advance \reqcolroom \ht\currbox
```

```
<*trace>
```

```
\fl@trace{float size = \the \ht \currbox (addtocurcol)}%

```

```
1105
```

```
\fl@trace{colroom = \the \colroom (addtocurcol)}%

```

```
1106
```

```
\fl@trace{reqcolroom = \the \reqcolroom (addtocurcol)}%

```

```
</trace>
```

```
\ifdim \colroom > \reqcolroom
```

```
\@fsetnum \colnum
```

```
1110
```

```
\ifnum \colnum > \z@
```

```
1111
```

```
\@bitor \currtype \deferlist
```

We need to defer the float also if its width doesn't fit.

```

1112
```

```
\@testwidth\currbox
```

```
<*trace>
```

```
\fl@trace{deferlist: \deferlist: (addtocurcol-before)}%

```

```
</trace>
```

```
\if@test
```

```
<*trace>
```

```
\fl@trace{type already on list: defer (addtocurcol)}%

```

```
</trace>
```

```
\else
```

```
\@bitor \currtype \botlist
```

```
<*trace>
```

```
\fl@trace{botlist: \botlist: (addtocurcol-before)}%

```

```
</trace>
```

```
\if@test
```

```
<*trace>
```

```

1127 \fl@trace{type already on list: bot---sent to addtobot}%
1128
```

```
1129 \@addtobot
```

```
1130 \else
```

```
1131
```

```
1132 \fl@trace{fpstype \ifodd \tempcnta OK \else not \fi
1133 here: \the \fpstype}%
```

```
1134
```

```
1135 \ifodd \count\@currbox
```

```
1136 \advance \reqcolroom \intextsep
```

```
1137 \ifdim \colroom>\reqcolroom
```

```
1138 \global \advance \colnum \m@ne
```

```
1139 \global \advance \textfloatsheight \ht\@currbox
```

This may sometimes give an overestimate.

```

1140 \global \advance \textfloatsheight 2\intextsep
1141 \cons \midlist \currbox

```

```
1142
```

```
1143 \fl@trace{***Success: here}%

```

```
1144 \fl@trace{textfloatsheight (after-here) =
1145 \the \textfloatsheight}%
1146 \fl@trace{colnum (after-here) = \the \colnum}%

```

```
1147
```

CHANGE TO \addtocurcol:

\penalty\z@ changed to \penalty\interlinepenalty so \samepage works properly with figure and table environments. (Changed 23 Oct 86)

There is also an \addpenalty\interlinepenalty above.

Since in 2e \samepage is no longer supported, these could be removed.

Although it is best to use \advspace in case two h floats come together, this makes other spacing more difficult to adjust; whereas if a user specifies two h floats together then they can more easily get the spacing correct by ad hoc commands.

It is necessary to adjust for the addition of \parskip here in case the float is added between paragraphs (i.e. when in vertical mode).

If the nobreak switch is true we need to reset it and clear \everypar since the float may not reset the flag and cannot reset the \everypar globally.

Typesetting starts here (we are in vertical mode).

```

1148 \if@nobreak
1149 \nobreak
1150 \nobreakfalse
1151 \everypar{}%
1152 \else
1153 \addpenalty \interlinepenalty
1154 \fi
1155 \vskip \intextsep
1156 \box\currbox
1157 \penalty\interlinepenalty
1158 \vskip\intextsep
1159 \ifnum\outputpenalty <-@\Mi \vskip -\parskip\fi

```

Typesetting ends here.

```

1160 \outputpenalty \z@
1161 \inserttrue

```

```
1162
```

```

1163 \else
1164 \fl@trace{Fail---no room at 2nd test of colroom
1165 (addtocorcol \string\intextsep)}%
1166
```

Next set of docstrip guards are a bit weird, essentially \@addtotoporbot ends up inside the kernel and the `fltrace` package and \@addtobot shows up in the `flafter` package. Guess that could have been done a bit more obvious :-)

```

1171 <{*2ekernel | fltrace | latexrelease}
1172 {*trace}
1173
```

```
1174
```

```
1175
```

```
1176 //2ekernel | fltrace | latexrelease)
1177 <!2ekernel&!fltrace&!latexrelease)
1178 {*trace}
1179
```

```
1180
```

```
1181
```

```
1182 //!2ekernel&!fltrace&!latexrelease)
1183
```

```
1184
```

```
1185
```

```
1186 {*trace}
1187
```

```
1188
```

```
1189 fpstype \the \@fpstype=ORD?}%
1190
```

```
1191 \ifnum \@fpstype<\sixt@n
1192
```

```
1193
```

```
1194
```

```
1195 {*trace}
1196
```

```
1197
```

```
1198
```

```
(addtocurcol)}%
```

```
1199
```

```
1200
```

```
1201
```

```
1202
```

```
1203
```

```
1204
```

```
1205
```

```
1206 {*trace}
1207
```

```
1208
```

```
1209
```

```
1210 {*trace}
1211
```

```
1212
```

```

1213 \fi
1214 }%
1215 </2ekernel | latexrelease | fltrace | flafter>
1216 <latexrelease | fltrace | flafter>\EndIncludeInRelease
1217 <latexrelease | fltrace | flafter>\IncludeInRelease{0000/00/00}%
1218 <latexrelease | fltrace | flafter> {\@addtocurcol}{float order in 2-column}%
1219 <latexrelease | fltrace | flafter>\def \@addtocurcol {%
1220 *<trace>
1221 <latexrelease | fltrace | flafter> \fl@trace{***Start addtocurcol}%
1222 /<trace>
1223 <latexrelease | fltrace | flafter> \Qinsertfalse
1224 <latexrelease | fltrace | flafter> \Qsetfloattypecounts
1225 <latexrelease | fltrace | flafter> \ifnum \Qfpstype=8
1226 *<trace>
1227 <latexrelease | fltrace | flafter> \fl@trace{fpstype !p only (addtocurcol):
1228 <latexrelease | fltrace | flafter> \the \Qfpstype = 8?}%
1229 /<trace>
1230 <latexrelease | fltrace | flafter> \else
1231 <latexrelease | fltrace | flafter> \ifnum \Qfpstype=24
1232 *<trace>
1233 <latexrelease | fltrace | flafter> \fl@trace{fpstype p only (addtocurcol):
1234 <latexrelease | fltrace | flafter> \the \Qfpstype = 24?}%
1235 /<trace>
1236 <latexrelease | fltrace | flafter> \else
1237 <latexrelease | fltrace | flafter> \Qflsettextmin

```

This is a new adjustment which is quite a major change in functionality; but it implements the documentation. Note that \reqcolroom will include the whole of the page-so-far, and hence includes \textfloatsheight of floats, so before comparing it with \textmin, we add this to \textmin also.

```

1238 *<trace>
1239 <latexrelease | fltrace | flafter> \fl@trace{textfloatsheight (before) =
1240 <latexrelease | fltrace | flafter> \the \textfloatsheight}%
1241 /<trace>
1242 <latexrelease | fltrace | flafter> \advance \textmin \textfloatsheight
1243 <latexrelease | fltrace | flafter> \reqcolroom \pageht

```

This line must be removed since \specialoutput changed.

```

1244 % \advance \reqcolroom \pagedp
1245 *<trace>
1246 <latexrelease | fltrace | flafter> \fl@trace{textmin + textfloatsheight:
1247 <latexrelease | fltrace | flafter> \the \textmin}%
1248 <latexrelease | fltrace | flafter> \fl@trace{page-so-far: \the \reqcolroom}%
1249 <latexrelease | fltrace | flafter>
1250 /<trace>
1251 <latexrelease | fltrace | flafter>
1252 <latexrelease | fltrace | flafter>
1253 *<trace>
1254 <latexrelease | fltrace | flafter> \ifdim \textmin>\reqcolroom
1255 /<trace>
1256 <latexrelease | fltrace | flafter> \reqcolroom \textmin
1257 <latexrelease | fltrace | flafter> \fl@trace{ORD? textmin being used}%
1258 *<trace>
1259 <latexrelease | fltrace | flafter> \fi
1260 <latexrelease | fltrace | flafter> \advance \reqcolroom \ht\currbox
1261 *<trace>
1262 <latexrelease | fltrace | flafter> \fl@trace{float size =
1263 <latexrelease | fltrace | flafter> \the \ht \currbox (addtocurcol)}%

```

```

1261 <{latexrelease | fltrace | flafter}>
1262 <{latexrelease | fltrace | flafter}>
1263 <{latexrelease | fltrace | flafter}>
1264 <{latexrelease | fltrace | flafter}>
1265 </trace>
1266 <{latexrelease | fltrace | flafter}>
1267 <{latexrelease | fltrace | flafter}>
1268 <{latexrelease | fltrace | flafter}>
1269 <{latexrelease | fltrace | flafter}>
1270 <{*trace}>
1271 <{latexrelease | fltrace | flafter}>
1272 <{latexrelease | fltrace | flafter}>
1273 </trace>
1274 <{latexrelease | fltrace | flafter}>
1275 <{*trace}>
1276 <{latexrelease | fltrace | flafter}>
1277 <{latexrelease | fltrace | flafter}>
1278 </trace>
1279 <{latexrelease | fltrace | flafter}>
1280 <{latexrelease | fltrace | flafter}>
1281 <{*trace}>
1282 <{latexrelease | fltrace | flafter}>
1283 <{latexrelease | fltrace | flafter}>
1284 </trace>
1285 <{latexrelease | fltrace | flafter}>
1286 <{*trace}>
1287 <{latexrelease | fltrace | flafter}>
1288 <{latexrelease | fltrace | flafter}>
1289 </trace>
1290 <{latexrelease | fltrace | flafter}>
1291 <{latexrelease | fltrace | flafter}>
1292 <{*trace}>
1293 <{latexrelease | fltrace | flafter}>
1294 <{latexrelease | fltrace | flafter}>
1295 <{latexrelease | fltrace | flafter}>
1296 </trace>
1297 <{latexrelease | fltrace | flafter}>
1298 <{latexrelease | fltrace | flafter}>
1299 <{latexrelease | fltrace | flafter}>
1300 <{latexrelease | fltrace | flafter}>
1301 <{latexrelease | fltrace | flafter}>
1302 <{latexrelease | fltrace | flafter}>

This may sometimes give an overestimate.

1303 <{latexrelease | fltrace | flafter}>
1304 <{latexrelease | fltrace | flafter}>
1305 <{latexrelease | fltrace | flafter}>
1306 <{*trace}>
1307 <{latexrelease | fltrace | flafter}>
1308 <{latexrelease | fltrace | flafter}>
1309 <{latexrelease | fltrace | flafter}>
1310 <{latexrelease | fltrace | flafter}>
1311 <{latexrelease | fltrace | flafter}>
1312 <{latexrelease | fltrace | flafter}>
1313 </trace>

\fl@trace{colroom =
 \the \@colroom (addtocurcol)}%
\fl@trace{reqcolroom =
 \the \@reqcolroom (addtocurcol)}%
\ifdim \@colroom>\@reqcolroom
 \f@lsetnum \@colnum
 \ifnum \@colnum>\z@
 \b@itor\@currtype\@deferlist
 \fl@trace{deferlist:
 \@deferlist: (addtocurcol-before)}%
 \if@test
 \fl@trace{type already on list:
 defer (addtocurcol)}%
 \else
 \b@itor\@currtype\@botlist
 \fl@trace{botlist: \@botlist:
 (addtocurcol-before)}%
 \if@test
 \fl@trace{type already on list:
 bot---sent to addtobot}%
 \else
 \f@l@trace{fpstype
 \ifodd \@tempcnta OK \else not \fi
 here: \the \@fpstype}%
 \ifodd \count\@currbox
 \advance \@reqcolroom \intextsep
 \ifdim \@colroom>\@reqcolroom
 \global \advance \@colnum \m@ne
 \global \advance
 \@textfloatsheight\ht\@currbox
 \global \advance
 \@textfloatsheight 2\intextsep
 \cons \@midlist \@currbox
 \f@l@trace{***Success: here}%
 \f@l@trace{textfloatsheight
 (after-here) =
 \the \@textfloatsheight}%
 \f@l@trace{colnum (after-here) =
 \the \@colnum}%

```

CHANGE TO \addtocurcol:  
\penalty\z@ changed to \penalty\interlinepenalty so \samepage works properly with figure and table environments. (Changed 23 Oct 86)

There is also an \addpenalty\interlinepenalty above.

Since in 2e \samepage is no longer supported, these could be removed.

Although it is best to use \addvspace in case two h floats come together, this makes other spacing more difficult to adjust; whereas if a user specifies two h floats together then they can more easily get the spacing correct by ad hoc commands.

It is necessary to adjust for the addition of \parskip here in case the float is added between paragraphs (i.e. when in vertical mode).

If the nobreak switch is true we need to reset it and clear \everypar since the float may not reset the flag and cannot reset the \everypar globally.

Typesetting starts here (we are in vertical mode).

```

1314 <(latexrelease | fltrace | flafter> \if@nobreak
1315 <(latexrelease | fltrace | flafter> \nobreak
1316 <(latexrelease | fltrace | flafter> \nobreakfalse
1317 <(latexrelease | fltrace | flafter> \everypar{}%
1318 <(latexrelease | fltrace | flafter> \else
1319 <(latexrelease | fltrace | flafter> \addpenalty\interlinepenalty
1320 <(latexrelease | fltrace | flafter> \fi
1321 <(latexrelease | fltrace | flafter> \vskip \intextsep
1322 <(latexrelease | fltrace | flafter> \box@\currbox
1323 <(latexrelease | fltrace | flafter> \penalty\interlinepenalty
1324 <(latexrelease | fltrace | flafter> \vskip\intextsep
1325 <(latexrelease | fltrace | flafter> \ifnum\outputpenalty
1326 <(latexrelease | fltrace | flafter> <- \OMii \vskip
1327 <(latexrelease | fltrace | flafter> -\parskip\fi

```

Typesetting ends here.

```

1328 <(latexrelease | fltrace | flafter> \outputpenalty \z@
1329 <(latexrelease | fltrace | flafter> \@inserttrue
1330 <*trace>
1331 <(latexrelease | fltrace | flafter> \else
1332 <(latexrelease | fltrace | flafter> \fl@trace{Fail---no room at 2nd test of colroom
1333 <(latexrelease | fltrace | flafter> (addtocorcol \string\intextsep)}%
1334 </trace>
1335 <(latexrelease | fltrace | flafter> \fi
1336 <(latexrelease | fltrace | flafter> \fi
1337 <(latexrelease | fltrace | flafter> \if@insert
1338 <(latexrelease | fltrace | flafter> \else

```

Next set of docstrip guards are a bit weird, essentially \addtotoporbot ends up inside the kernel and the fltrace package and \addtotoporbot shows up in the flafter package. Guess that could have been done a bit more obvious :-)

```

1339 <*2ekernel | fltrace>
1340 <*trace>
1341 <(latexrelease | fltrace | flafter> \fl@trace{not here: sent to addtotoporbot}%
1342 </trace>
1343 <(latexrelease | fltrace | flafter> \addtotoporbot
1344 </2ekernel | fltrace>
1345 <!*2ekernel&!autoload&!fltrace>
1346 <*trace>
1347 <(latexrelease | fltrace | flafter> \fl@trace{not here: sent to addtobot}%

```

```

1348 </trace>
1349 <|latexrelease | fltrace | flafter> \@addtobot
1350 <|!2ekernel&!autoload&!fltrace>
1351 <|latexrelease | fltrace | flafter> \fi
1352 <|latexrelease | fltrace | flafter> \fi
1353 <|latexrelease | fltrace | flafter> \fi
1354 <*trace>
1355 <|latexrelease | fltrace | flafter> \else
1356 <|latexrelease | fltrace | flafter> \fl@trace{Fail: colnum = \the \@colnum:
1357 <|latexrelease | fltrace | flafter> fpstype \the \@fpstype=ORD?}%
1358 <|latexrelease | fltrace | flafter> \ifnum \@fpstype<\sixt@on
1359 <|latexrelease | fltrace | flafter> \fl@trace{ERROR: BANG float not successful
1360 <|latexrelease | fltrace | flafter> (addtocurcol)}%
1361 <|latexrelease | fltrace | flafter> \fi
1362 </trace>
1363 <|latexrelease | fltrace | flafter> \fi
1364 <*trace>
1365 <|latexrelease | fltrace | flafter> \else
1366 <|latexrelease | fltrace | flafter> \fl@trace{Fail---no room: fl box ht:
1367 <|latexrelease | fltrace | flafter> \the \ht \currbox (addtocurcol)}%
1368 </trace>
1369 <|latexrelease | fltrace | flafter> \fi
1370 <|latexrelease | fltrace | flafter> \fi
1371 <|latexrelease | fltrace | flafter> \fi
1372 <|latexrelease | fltrace | flafter> \if@insert
1373 <|latexrelease | fltrace | flafter> \else
1374 <|latexrelease | fltrace | flafter> \resethfps
1375 <*trace>
1376 <|latexrelease | fltrace | flafter> \fl@trace{put on deferlist (addtocurcol)}%
1377 </trace>
1378 <|latexrelease | fltrace | flafter> \@cons\@deferlist\currbox
1379 <*trace>
1380 <|latexrelease | fltrace | flafter> \fl@trace{deferlist: \@deferlist:
1381 <|latexrelease | fltrace | flafter> (addtocurcol-after)}%
1382 </trace>
1383 <|latexrelease | fltrace | flafter> \fi
1384 <|latexrelease | fltrace | flafter> }%
1385 <|latexrelease | fltrace | flafter>\EndIncludeInRelease

```

(End definition for \@addtocurcol.)

\@addtonextcol Lots of changes.

```

1386 <|latexrelease | fltrace>\IncludeInRelease{2015/01/01}
1387 <|latexrelease | fltrace> {\@addtonextcol}{float order in 2-column}%
1388 <|2ekernel | latexrelease | fltrace>
1389 \def\@addtonextcol{%
1390 \begingroup
1391 <*trace>
1392 \fl@trace{***Start addtonextcol}%
1393 </trace>
1394 \insertfalse
1395 \setfloattypecounts
1396 \ifnum \@fpstype=8
1397 <*trace>

```

```

1398 \f1@trace{fpstype not curcol: \the \@fpstype = 8?}%
1399 </trace>
1400 \else
1401 \ifnum \@fpstype=24
1402 <*trace>
1403 \f1@trace{fpstype not curcol: \the \@fpstype = 24?}%
1404 </trace>
1405 \else
1406 \o@f1settextmin
1407 <*trace>
1408 \f1@trace{text-so-far: Opt (top of col)}%
1409 </trace>
1410 \o@reqcolroom \ht\@currbox
1411 <*trace>
1412 \f1@trace{float size: \the \o@reqcolroom (addtonextcol)}%
1413 </trace>
1414 \advance \o@reqcolroom \o@textmin
1415 <*trace>
1416 \f1@trace{colroom = \the \o@colroom (addtonextcol)}%
1417 \f1@trace{reqcolroom = \the \o@reqcolroom (addtonextcol)}%
1418 </trace>
1419 \ifdim \o@colroom>\o@reqcolroom
1420 \o@f1setnum \o@colnum
1421 \ifnum\o@colnum>\z@
1422 \o@bito\o@currtype\o@deferlist
1423 <*trace>
1424 \f1@trace{deferlist: \o@deferlist: (addtonextcol-before)}%
1425 </trace>
1426 \o@testwidth\o@currbox
1427 <*trace>
1428 </trace>
1429 \f1@trace{type already on list: defer (addtonextcol)}%
1430 </trace>
1431 \else
1432 <*trace>
1433 \f1@trace{sent to addtotoporbot (addtonextcol)}%
1434 </trace>
1435 \o@addtotoporbot
1436 \fi
1437 <*trace>
1438 </trace>
1439 \else
1440 \f1@trace{Fail---no room: fl box ht: \the \ht \o@currbox
1441 (addtonextcol)}%
1442 </trace>
1443 \fi
1444 \fi
1445 \fi
1446 \o@if@insert
1447 \else
1448 <*trace>
1449 \f1@trace{put back on deferlist (addtonextcol)}%
1450 </trace>

```

```

1451 \@cons\@deferlist\@currbox
1452 <*trace>
1453 \fl@trace{deferlist: \@deferlist: (addtonextcol-after)}%
1454 </trace>
1455 \fi
1456 <*trace>
1457 \fl@trace{End of addtonextcol -- locally counts:}%
1458 \fl@trace{col: \the\@colnum. top: \the \@topnum. bot: \the \@botnum.}%
1459 </trace>
1460 \endgroup
1461 <*trace>
1462 \fl@trace{End of addtonextcol -- globally counts:}%
1463 \fl@trace{col: \the\@colnum. top: \the \@topnum. bot: \the \@botnum.}%
1464 </trace>
1465 }%
1466 </2ekernel | latexrelease | fltrace>
1467 <latexrelease | fltrace>\EndIncludeInRelease
1468 <latexrelease | fltrace>\IncludeInRelease{0000/00/00}%
1469 <latexrelease | fltrace> {\@addtonextcol}{float order in 2-column}%
1470 <latexrelease | fltrace>\def\@addtonextcol{%
1471 <latexrelease | fltrace> \begingroup
1472 <*trace>
1473 <latexrelease | fltrace> \fl@trace{***Start addtonextcol}%
1474 </trace>
1475 <latexrelease | fltrace> \@insertfalse
1476 <latexrelease | fltrace> \@setfloattypecounts
1477 <latexrelease | fltrace> \ifnum \fpstype=8
1478 <*trace>
1479 <latexrelease | fltrace> \fl@trace{fpstype not curcol:
1480 <latexrelease | fltrace> \the \fpstype = 8?}%
1481 </trace>
1482 <latexrelease | fltrace> \else
1483 <latexrelease | fltrace> \ifnum \fpstype=24
1484 <*trace>
1485 <latexrelease | fltrace> \fl@trace{fpstype not curcol:
1486 <latexrelease | fltrace> \the \fpstype = 24?}%
1487 </trace>
1488 <latexrelease | fltrace> \else
1489 <latexrelease | fltrace> \fsettextmin
1490 <*trace>
1491 <latexrelease | fltrace> \fl@trace{text-so-far: Opt (top of col)}%
1492 </trace>
1493 <latexrelease | fltrace> \reqcolroom \ht\currbox
1494 <*trace>
1495 <latexrelease | fltrace> \fl@trace{float size:
1496 <latexrelease | fltrace> \the \reqcolroom (addtonextcol)}%
1497 <latexrelease | fltrace>
1498 </trace>
1499 <latexrelease | fltrace> \advance \reqcolroom \textmin
1500 <*trace>
1501 <latexrelease | fltrace> \fl@trace{colroom =
1502 <latexrelease | fltrace> \the \colroom (addtonextcol)}%
1503 <latexrelease | fltrace> \fl@trace{reqcolroom =
1504 <latexrelease | fltrace> \the \reqcolroom (addtonextcol)}%

```

```

1505 </trace>
1506 <|latexrelease | fltrace> \ifdim \@colroom>\@reqcolroom
1507 <|latexrelease | fltrace> \@flsetnum \@colnum
1508 <|latexrelease | fltrace> \ifnum\@colnum>\z@
1509 <|latexrelease | fltrace> \@bitor\@currtype\@deferlist
1510 <|*trace> \f@trace{deferlist: \@deferlist:
1511 <|latexrelease | fltrace> (addtonextcol-before)}%
1512 <|latexrelease | fltrace>
1513 </trace>
1514 <|latexrelease | fltrace> \if@test
1515 <|*trace> \f@trace{type already on list:
1516 <|latexrelease | fltrace> defer (addtonextcol)}%
1517 <|latexrelease | fltrace>
1518 </trace>
1519 <|latexrelease | fltrace> \else
1520 <|*trace> \f@trace{sent to addtotopbot
1521 <|latexrelease | fltrace> (addtonextcol)}%
1522 <|latexrelease | fltrace>
1523 </trace>
1524 <|latexrelease | fltrace> \addtotopbot
1525 <|latexrelease | fltrace> \fi
1526 <|latexrelease | fltrace> \fi
1527 <|*trace> \else
1528 <|latexrelease | fltrace> \f@trace{Fail---no room: fl box ht:
1529 <|latexrelease | fltrace> \the \ht \@currbox (addtonextcol)}%
1530 <|latexrelease | fltrace>
1531 </trace> \fi
1532 <|latexrelease | fltrace> \fi
1533 <|latexrelease | fltrace> \fi
1534 <|latexrelease | fltrace> \if@insert
1535 <|latexrelease | fltrace> \else
1536 <|latexrelease | fltrace> \f@trace{put back on deferlist
1537 <|*trace> (addtonextcol)}%
1538 <|latexrelease | fltrace>
1539 <|latexrelease | fltrace>
1540 </trace>
1541 <|latexrelease | fltrace> \cons\@deferlist\@currbox
1542 <|*trace> \f@trace{deferlist: \@deferlist:
1543 <|latexrelease | fltrace> (addtonextcol-after)}%
1544 <|latexrelease | fltrace>
1545 </trace>
1546 <|latexrelease | fltrace> \fi
1547 <|*trace> \f@trace{End of addtonextcol --
1548 <|latexrelease | fltrace> locally counts:}%
1549 <|latexrelease | fltrace> \f@trace{col: \the \@colnum.
1550 <|latexrelease | fltrace> top: \the \@topnum. bot: \the \@botnum.}%
1551 <|latexrelease | fltrace>
1552 </trace>
1553 <|latexrelease | fltrace> \endgroup
1554 <|*trace> \f@trace{End of addtonextcol --
1555 <|latexrelease | fltrace> globally counts:}%
1556 <|latexrelease | fltrace> \f@trace{col: \the \@colnum.
1557 <|latexrelease | fltrace> top: \the \@topnum. bot: \the \@botnum.}%
1558 <|latexrelease | fltrace>

```

```

1559 </trace>
1560 <|latexrelease | fltrace}>}%
1561 <|latexrelease | fltrace}>\EndIncludeInRelease

(End definition for \@addtonextcol.)
```

\@addtobdblcol Lots of changes.

```

1562 <|latexrelease | fltrace}>\IncludeInRelease{2015/01/01}%
1563 <|latexrelease | fltrace}> {\@addtobdblcol}{float order in 2-column}%
1564 {*2ekernel | latexrelease | fltrace}
1565 \def\@addtobdblcol{%
1566 \begingroup
1567 <*trace>
1568 \fl@trace{***Start addtobdblcol}%
1569 </trace>
1570 \atinsertfalse
1571 \setfloattypecounts
1572 \getfpsbit \tw@
1573 <*trace>
1574 \fl@trace{fpstype \ifodd \tempcnta OK \else not \fi dbltop:
1575 \the \fpstype}%
1576 </trace>
1577 \ifodd\tempcnta
1578 \iflsetnum \dbltopnum
1579 \ifnum \dbltopnum>\z@
1580 \tempswafalse
1581 \ifdim \dbltoproom>\ht\currbox
1582 \tempswatrue
1583 <*trace>
1584 \fl@trace{Space OK: \dbltoproom =
1585 \the \dbltoproom > \the \ht \currbox
1586 (\dbltoproom)}%
1587 </trace>
1588 \else
1589 <*trace>
1590 \fl@trace{fpstype: \the \fpstype (addtobdblcol)}%
1591 </trace>
1592 \ifnum \fpstype<\sixt@n
1593 <*trace>
1594 \fl@trace{BANG float ignoring \dbltoproom}%
1595 \fl@trace{@spaces \dbltoproom = \the \dbltoproom.
1596 Ht float: \the \ht \currbox-BANG}%
1597 </trace>

```

Need to check that there is room on the page, using the local value of \textmin to make the necessary adjustment to \dbltoproom.

```

1598 \advance \dbltoproom \textmin
1599 <*trace>
1600 \fl@trace{Local value of texmin: \the\textmin}%
1601 \fl@trace{@spaces space on page = \the \dbltoproom.
1602 Ht float: \the \ht \currbox-BANG}%
1603 </trace>
1604 \ifdim \dbltoproom>\ht\currbox
1605 \tempswatrue

```

```

1606 <*trace>
1607 \fl@trace{Space OK BANG: space on page =
1608 \the \dbltoproom > \the \ht \currbox}%
1609 \else
1610 \fl@trace{fpstype: \the \fpstype}%
1611 \fl@trace{Fail---no room dbltoproom-BANG?:}%
1612 \fl@trace{\spaces space on page = \the \dbltoproom.
1613 Ht float: \the \ht \currbox}%
1614 </trace>
1615 \fi
1616 \advance \dbltoproom -\textmin
1617 <*trace>
1618 \else
1619 \fl@trace{fpstype: \the \fpstype}%
1620 \fl@trace{Fail---no room dbltoproom-ORD?:}%
1621 \fl@trace{\spaces \dbltoproom = \the \dbltoproom.
1622 Ht float: \the \ht \currbox}%
1623 </trace>
1624 \fi
1625 \fi
1626 \if@tempswa
1627 \bitor \currtype \deferlist
1628 <*trace>
1629 \fl@trace{(dbl)deferlist: \deferlist: (before)}%
1630 </trace>
not in fixfloats?
1631 \testwrongwidth\currbox
1632 \if@test
1633 <*trace>
1634 \fl@trace{type already on list: (dbl)defer}%
1635 </trace>
1636 \else
1637 \tempdima -\ht\currbox
1638 \advance\tempdima
1639 -\ifx \dbltoplist\empty \dbltextfloatsep \else
1640 \dblfloatsep \fi
1641 \global \advance \dbltoproom \tempdima
1642 \global \advance \colht \tempdima
1643 \global \advance \dbltopnum \m@ne
1644 \cons \dbltoplist \currbox
1645 <*trace>
1646 \fl@trace{dbltopnum (after) = \the \dbltopnum}%
1647 \fl@trace{***Success: dbltop}%
1648 </trace>
1649 \inserttrue
1650 \fi
1651 \fi
1652 <*trace>
1653 \else
1654 \fl@trace{Fail: dbltopnum = \the \dbltopnum: fpstype
1655 \the \fpstype=ORD?}%
1656 \ifnum \fpstype<\six@on
1657 \fl@trace{ERROR: !t float not successful (addtoblcol)}%

```

```

1658 \fi
1659 </trace>
1660 \fi
1661 \fi
1662 \if@insert
1663 \else
1664 <*trace>
1665 \fl@trace{put on deferlist}%
1666 </trace>
1667 \@cons\@deferlist\@currbox
1668 <*trace>
1669 \fl@trace{(dbl)deferlist: \@deferlist: (after)}%
1670 </trace>
1671 \fi
1672 <*trace>
1673 \fl@trace{End of addtobtblcol -- locally count:}%
1674 \fl@trace{ dbltop: \the \dbltopnum.}%
1675 </trace>
1676 \endgroup
1677 <*trace>
1678 \fl@trace{End of addtobtblcol -- globally count:}%
1679 \fl@trace{dbltop: \the \dbltopnum.}%
1680 </trace>
1681 }%
1682 </2ekernel | latexrelease | fltrace>
1683 <latexrelease | fltrace>\EndIncludeInRelease
1684 <latexrelease | fltrace>\IncludeInRelease[0000/00/00]%
1685 <latexrelease | fltrace> {\@addtobtblcol}{float order in 2-column}%
1686 <latexrelease | fltrace>\def\@addtobtblcol{%
1687 <latexrelease | fltrace> \begingroup
1688 <*trace>
1689 <latexrelease | fltrace> \fl@trace{***Start addtobtblcol}%
1690 </trace>
1691 <latexrelease | fltrace> \insertfalse
1692 <latexrelease | fltrace> \setfloattypecounts
1693 <latexrelease | fltrace> \getfpsbit \tw@
1694 <*trace>
1695 <latexrelease | fltrace> \fl@trace{fpstype \ifodd \tempcnta OK
1696 <latexrelease | fltrace> \else not \fi dbltop: \the \fpstype}%
1697 </trace>
1698 <latexrelease | fltrace> \ifodd\tempcnta
1699 <latexrelease | fltrace> \flsetnum \dbltopnum
1700 <latexrelease | fltrace> \ifnum \dbltopnum>\z@
1701 <latexrelease | fltrace> \tempswafalse
1702 <latexrelease | fltrace> \ifdim \dbltoproom>\ht\currbox
1703 <latexrelease | fltrace> \tempswatrue
1704 <*trace>
1705 <latexrelease | fltrace> \fl@trace{Space OK: \dbltoproom =
1706 <latexrelease | fltrace> \the \dbltoproom > \the \ht \currbox
1707 <latexrelease | fltrace> (dbltoproom)}%
1708 </trace>
1709 <latexrelease | fltrace> \else
1710 <*trace>
1711 <latexrelease | fltrace> \fl@trace{fpstype: \the \fpstype (addtobtblcol)}%

```

```

1712 </trace>
1713 <latexrelease | fltrace> \ifnum \@fpstype<\sixt@n
1714 <*trace>
1715 <latexrelease | fltrace> \fl@trace{BANG float ignoring \@dbltoproom}%
1716 <latexrelease | fltrace> \fl@trace{\@spaces \@dbltoproom =
1717 <latexrelease | fltrace> \the \@dbltoproom.
1718 <latexrelease | fltrace> Ht float: \the \ht \currbox-BANG}%
1719 </trace>

```

Need to check that there is room on the page, using the local value of \textmin to make the necessary adjustment to \@dbltoproom.

```

1720 <latexrelease | fltrace> \advance \@dbltoproom \textmin
1721 <*trace>
1722 <latexrelease | fltrace> \fl@trace{Local value of texmin: \the\textmin}%
1723 <latexrelease | fltrace> \fl@trace{\@spaces space on page =
1724 <latexrelease | fltrace> \the \@dbltoproom.
1725 <latexrelease | fltrace> Ht float: \the \ht \currbox-BANG}%
1726 </trace>
1727 <latexrelease | fltrace>
1728 <latexrelease | fltrace>
1729 <*trace>
1730 <latexrelease | fltrace>
1731 <latexrelease | fltrace>
1732 <latexrelease | fltrace>
1733 <latexrelease | fltrace>
1734 <latexrelease | fltrace>
1735 <latexrelease | fltrace>
1736 <latexrelease | fltrace>
1737 <latexrelease | fltrace>
1738 </trace>
1739 <latexrelease | fltrace>
1740 <latexrelease | fltrace>
1741 <*trace>
1742 <latexrelease | fltrace>
1743 <latexrelease | fltrace>
1744 <latexrelease | fltrace>
1745 <latexrelease | fltrace>
1746 <latexrelease | fltrace>
1747 <latexrelease | fltrace>
1748 </trace>
1749 <latexrelease | fltrace>
1750 <latexrelease | fltrace>
1751 <latexrelease | fltrace>
1752 <latexrelease | fltrace>
1753 <*trace>
1754 <latexrelease | fltrace>
1755 <latexrelease | fltrace>
1756 </trace>
1757 <latexrelease | fltrace>
1758 <*trace>
1759 <latexrelease | fltrace>
1760 </trace>
1761 <latexrelease | fltrace>
1762 <latexrelease | fltrace>

```

```

\ifdim \@dbltoproom>\ht\currbox
\@tempswatrue
\fl@trace{Space OK BANG: space on page =
\the\@dbltoproom > \the\ht\currbox}%
\else
\fl@trace{fpstype: \the \@fpstype}%
\fl@trace{Fail---no room dbltoproom-BANG?:}%
\fl@trace{\@spaces space on page =
\the \@dbltoproom.
Ht float: \the \ht \currbox}%
\fi
\advance \@dbltoproom -\textmin
\else
\fl@trace{fpstype: \the \@fpstype}%
\fl@trace{Fail---no room dbltoproom-ORD?:}%
\fl@trace{\@spaces \@dbltoproom =
\the \@dbltoproom.
Ht float: \the \ht \currbox}%
\fi
\@if@tempswa
\@bitor \currtype \@dbldeferlist
\fl@trace{dbldeferlist:
@dbldeferlist: (before)}%
\@if@test
\fl@trace{type already on list: dbldefer}%
\else
\@tempdima -\ht\currbox

```

```

1763 ⟨latexrelease | fltrace⟩
1764 ⟨latexrelease | fltrace⟩
1765 ⟨latexrelease | fltrace⟩
1766 ⟨latexrelease | fltrace⟩
1767 ⟨latexrelease | fltrace⟩
1768 ⟨latexrelease | fltrace⟩
1769 ⟨latexrelease | fltrace⟩
1770 ⟨latexrelease | fltrace⟩
1771 ⟨*trace⟩
1772 ⟨latexrelease | fltrace⟩
1773 ⟨latexrelease | fltrace⟩
1774 ⟨latexrelease | fltrace⟩
1775 ⟨/trace⟩
1776 ⟨latexrelease | fltrace⟩
1777 ⟨latexrelease | fltrace⟩
1778 ⟨latexrelease | fltrace⟩
1779 ⟨*trace⟩
1780 ⟨latexrelease | fltrace⟩
1781 ⟨latexrelease | fltrace⟩
1782 ⟨latexrelease | fltrace⟩
1783 ⟨latexrelease | fltrace⟩
1784 ⟨latexrelease | fltrace⟩
1785 ⟨latexrelease | fltrace⟩
1786 ⟨latexrelease | fltrace⟩
1787 ⟨/trace⟩
1788 ⟨latexrelease | fltrace⟩
1789 ⟨latexrelease | fltrace⟩
1790 ⟨latexrelease | fltrace⟩
1791 ⟨latexrelease | fltrace⟩
1792 ⟨*trace⟩
1793 ⟨latexrelease | fltrace⟩
1794 ⟨/trace⟩
1795 ⟨latexrelease | fltrace⟩
1796 ⟨*trace⟩
1797 ⟨latexrelease | fltrace⟩
1798 ⟨/trace⟩
1799 ⟨latexrelease | fltrace⟩
1800 ⟨*trace⟩
1801 ⟨latexrelease | fltrace⟩
1802 ⟨latexrelease | fltrace⟩
1803 ⟨/trace⟩
1804 ⟨latexrelease | fltrace⟩
1805 ⟨*trace⟩
1806 ⟨latexrelease | fltrace⟩
1807 ⟨latexrelease | fltrace⟩
1808 ⟨/trace⟩
1809 ⟨latexrelease | fltrace⟩}%
1810 ⟨latexrelease | fltrace⟩\EndIncludeInRelease

(End definition for \@addtoblcol.)
```

\@addmarginpar

```

1811 ⟨*2ekernel⟩
1812 \def\@addmarginpar{\@next\@marbox\@currlist{\@cons\@freelist\@marbox}
```

```

1813 \@cons\@freelist\@currbox}\@latexbug\@tempcnta\@ne
1814 \if@twocolumn
1815 \if@firstcolumn \@tempcnta\m@ne \fi
1816 \else
1817 \if@mparswitch
1818 \ifodd\c@page \else\@tempcnta\m@ne \fi
1819 \fi
1820 \if@reversemargin \@tempcnta -\@tempcnta \fi
1821 \fi
1822 \ifnum\@tempcnta <\z@ \global\setbox\@marbox\box\@currbox \fi
1823 \global\@tempdima\@mparbottom
1824 \advance\@tempdima -\@pageht
1825 \advance\@tempdima\ht\@marbox
1826 \ifdim\@tempdima >\z@
1827 \@latex@warning@no@line {Marginpar on page \thepage\space moved}%
1828 \else
1829 \global\@tempdima\z@
1830 \fi
1831 \global\@mparbottom\@pageht
1832 \global\advance\@mparbottom\@tempdima
1833 \global\advance\@mparbottom\dp\@marbox
1834 \global\advance\@mparbottom\marginparpush
1835 \advance\@tempdima -\ht\@marbox

```

Putting box movement inside the ‘marbox’:

```

1836 \global\setbox\@marbox
1837 \vbox {\vskip\@tempdima
1838 \box\@marbox}%
1839 \global\ht\@marbox\z@
1840 \global\dp\@marbox\z@

```

Sticking (rather than gluing:-) the ‘marbox’ to the line above, changed vskip to kern:

```

1841 \kern -\@pagedp
1842 \nointerlineskip
1843 \hb@xt@\columnwidth
1844 {\ifnum\@tempcnta >\z@
1845 \hskip\columnwidth \hskip\marginparsep
1846 \else
1847 \hskip -\marginparsep \hskip -\marginparwidth
1848 \fi
1849 \box\@marbox\hss}%

```

For this reason the following code can vanish:

```

\nobreak %% No longer needed. CAR92/12
\vskip -\@tempdima %% No longer needed. CAR92/12
1850 \nointerlineskip
1851 \hbox{\vrule\height\z@ \width\z@ \depth\@pagedp}}

```

*(End definition for \@addmarginpar.)*

### 1.1.1 Kludgeins

This part of the file is part of the implementation of the following two new commands for L<sup>A</sup>T<sub>E</sub>X2e.

```
\enlargethispage{<dim>}
```

Adds `<dim>` to the height of the current column only. On the printed page the bottom of this column is extended downwards by exactly `<dim>` without having any effect on the placement of the footer; this may result in an overprinting.

```
\enlargethispage*{<dim>}
```

Similar to `\enlargethispage` but it tries to squeeze the column to be printed in as small a space as possible, ie it uses any shrinkability in the column. If the column was not explicitly broken (e.g. with `\pagebreak`) this may result in an overfull box message but except for this it will come out as expected (if you know what to expect).

The star form of this command is dedicated to Leslie Lamport, the other we need for ourselves (FMi, CAR).

These commands may well have unwanted effects if used soon before a `\clearpage`: please give keep them clear of such places.

`\@kludgeins` The insert which makes TeX do a lot of the necessary work. All we need to put into it is the amount by which the pagegoal should be changed.

```
1852 \newinsert \@kludgeins
1853 \global\dimen\@kludgeins \maxdimen
1854 \global\count\@kludgeins 1000
```

(End definition for `\@kludgeins`.)

`\enlargethispage` The user command.

```
1855 \gdef \enlargethispage {
1856 \@ifstar
1857 {
1858 (*trace)
1859 \f1@trace{Enlarging page height * }%
1860 (/trace)
1861 \enlargepage{\hbox{\kern\p@}}%
1862 {
1863 (*trace)
1864 \f1@trace{Enlarging page height exactly---}%
1865 (/trace)
1866 \enlargepage\empty%
1867 }
```

(End definition for `\enlargethispage` and `\enlargethispage*`.)

`\@enlargepage` This actually inserts the insert, after checking for extreme values of the change.

```
1868 \gdef\@enlargepage#1#2{
1869 (*trace)
1870 \f1@trace{\@spaces\@spaces by #2}%
1871 (/trace)
1872 \tempskipa#2\relax
1873 \ifdim \tempskipa>.5\maxdimen
1874 \@latex@error{Suggested\space extra\space height\space
1875 (the\tempskipa)\space dangerously\space
1876 large}\@eha
1877 \else
1878 \ifdim \vsize<.5\maxdimen
```

```

1879 {*trace}
1880 \fl@trace {Kludgeins added--pagegoal before: \the\pagegoal}%
1881
```

```

1882 \@bsphack
1883 \insert\@kludgeins{\#1\vskip-\@tempskipa}%
1884 \@esphack

```

This next bit is for tracing only:

```

1885 {*trace}
1886 \ifvmode \par
1887 \fl@trace {Kludgeins added--pagegoal after: \the \pagegoal}%
1888 \fi
1889
```

```

1890 \else
1891 \@latex@error{Page\space height\space already\space
1892 too\space large}\@eha

```

```

1893 \fi
1894
```

```

1895 }

```

(End definition for \enlargepage.)

### \ShowFloat

This command provides some information about the contents of a float register. Float registers have internal names of the form \bx@\langle Uppercase-letter(s)-or numbers\rangle and you specify just this letter or letters as the argument, e.g., \ShowFloat{A}. (There is not much error recovery if you specify something that isn't a float.)

```

1896
```

```

1897
```

```

1898
```

```

1899
```

```

1900
```

```

1901
```

```

1902
```

```

1903
```

```

1904
```

```

1905
```

```

1906
```

```

1907
```

```

1908
```

```

1909 }

```

```

1910
```

```

1911
```

```

1912
```

```

1913
```

```

1914
```

```

1915
```

```

1916 }

```

Here are two definitions from fltrace that make the above code work:

```

1917
```

```

1918
```

```

1919
```

```

1920
```

```

1921
```

```

1921 \langle latexrelease\rangle\IncludeInRelease{0000/00/00}%
1922 \langle latexrelease\rangle {\ShowFloat}{Show float register contents}%
1923 \langle latexrelease\rangle
1924 \langle latexrelease\rangle\let\ShowFloat\@undefined
1925 \langle latexrelease\rangle\let\f1@ShowFloat\@undefined
1926 \langle latexrelease\rangle\let\f1@traceval\@undefined
1927 \langle latexrelease\rangle\let\f1@tracemessage\@undefined
1928 \langle latexrelease\rangle\EndIncludeInRelease

```

(End definition for `\ShowFloat`.)

### 1.1.2 Float control

This part implements controllable floats and other changes to the float mechanism.

It provides, at the document level, the following command for inclusion in L<sup>A</sup>T<sub>E</sub>X2e.

`\suppressfloats`

This suppresses all further floats on the current page.

With an optional argument it suppresses only floats only in certain positions on the current page.

[**t**] suppresses only floats at the top of the page [**b**] suppresses only floats at the bottom of the page

It also enables the use of an extra specifier, **!**, in the location optional argument of a float. If this is present then, just for this particular float, whenever it is processed by the float mechanism the following are ignored:

- all restrictions on the number of floats which can appear;
- all explicit restrictions on the amount of space which should (not) be occupied by floats and/or text.

The mechanism will still attempt to ensure that pages are not overfull.

These specifiers override, for the single float, the suppression commands described above.

In its current form, it also supplies a reasonably exhaustive, and somewhat baroque, means of tracing some aspects of the float mechanism.

More tracing.

<code>\f1@trace</code>	Set-up tracing for floats independent of other tracing as it produces mega-output. Default is no tracing.
<code>\tracefloatoff</code>	
<code>\tracefloats</code>	1929 <code>(*f1trace)</code>
<code>\f1@traceval</code>	1930 <code>\def\f1@tracemessage #1{\{\let\@elt\@empty\typeout{LaTeX2e: #1}\}}</code>
<code>\tracefloatvals</code>	1931 <code>\def\tracefloats{\let\f1@trace\f1@tracemessage}</code>
<code>\f1@tracemessage</code>	1932 <code>\def\tracefloatoff{\let\f1@trace\@gobble}</code>
	1933 <code>\tracefloatoff</code>
	1934 <code>\def\f1@traceval #1{\f1@trace{\string#1 = \the#1}}</code>
	1935 <code>\IncludeInRelease{2015/01/01}{\tracefloatvals}%</code>
	1936 <code>{trace float vals}%</code>
	1937 <code>\def\tracefloatvals{%</code>

As `\@dblfloatplacement` sets `\f@depth` it needs to be run inside a group, otherwise the float placement will test for the wrong value.<sup>42</sup>

1938    \begingroup

When the user requests `\tracefloatvals` then they should show regardless of the tracing state, so locally we make sure that it is activated.

```
1939 \tracefloats
1940 \@dblfloatplacement
1941 \@floatplacement
1942 \f@trace{***Float placement parameters:}%
1943 \f@traceval\@colnum
1944 \f@traceval\@colroom
1945 \f@traceval\@topnum
1946 \f@traceval\@toproom
1947 \f@traceval\@botnum
1948 \f@traceval\@botroom
1949 \f@traceval\@fpmin
1950 \f@trace{\string\textration = \textfraction}%
1951 \f@traceval\@dbltopnum
1952 \f@traceval\@dbltoproom
1953 \f@tracef{\string\textration = \textfraction}%
1954 \f@trace{toplist: \@toplist}%
1955 \f@trace{botlist: \@botlist}%
1956 \f@trace{midlist: \@midlist}%
1957 \f@trace{deferlist: \@deferlist}%
1958 \f@trace{dbltoplist: \@dbltoplist}%
1959 %FMi \f@trace{dbldeferlist: \@dbldeferlist}%
1960 \endgroup
1961 }
1962 \EndIncludeInRelease
1963 \IncludeInRelease{0000/00/00}{\tracefloatvals}%
1964 {trace float vals}%
1965 \def \tracefloatvals{%
1966 \begingroup
1967 \tracefloats
1968 \@dblfloatplacement
1969 \@floatplacement
1970 \f@trace{***Float placement parameters:}%
1971 \f@traceval\@colnum
1972 \f@traceval\@colroom
1973 \f@traceval\@topnum
1974 \f@traceval\@toproom
1975 \f@traceval\@botnum
1976 \f@traceval\@botroom
1977 \f@traceval\@fpmin
1978 \f@trace{\string\textration = \textfraction}%
1979 \f@traceval\@dbltopnum
1980 \f@traceval\@dbltoproom
1981 \f@trace{\string\textration = \textfraction}%
1982 \f@trace{toplist: \@toplist}%
1983 \f@trace{botlist: \@botlist}%
1984 \f@trace{midlist: \@midlist}}
```

---

<sup>42</sup>This is a somewhat questionable design.

```

1985 \fl@trace{deferlist: \@deferlist}%
1986 \fl@trace{dbltoplist: \@dbltoplist}%
1987 % next line only in old releases
1988 \fl@trace{dbldeferlist: \@dbldeferlist}%
1989 \endgroup
1990 }
1991 \EndIncludeInRelease

```

We need to make sure that `fltrace` comes before `flafter` to make the tracing work.

```

1992 \@ifpackageloaded{flafter}
1993 { \PackageWarningNoLine
1994 {fltrace}{Load 'fltrace' before 'flafter'\MessageBreak
1995 Attempting to recover by reloading 'flafter'}%

```

Hide the fact that `flafter` was already loaded and then request it anew.

```

1996 \expandafter\let\csname ver@flafter.sty\endcsname\relax
1997 \def\reserved@a#1{%
1998 \expandafter\let\csname\string#1+flafter+IIR\endcsname\relax}%
1999 \reserved@a\@addtocurcol
2000 \reserved@a\@addtonextcol
2001 \RequirePackage{flafter}{}}
2002
```

As the code for `flafter` will contain tracing calls so that it works in conjunction with `fltrace` we need to provide a dummy definition for `\fl@trace` in that package.

```

2003 <*flafter>
2004 \providecommand\fl@trace[1]{}
2005
```

*(End definition for `\fl@trace` and others.)*

`\suppressfloats` Float suppression commands: these set the relevant counter globally to zero. Thus they  
`\@flstop` are overridden for a particular float by an ! specifier.

```

2006 <*2ekernel>
2007 \def \suppressfloats {%
2008 \ifnextchar [%
2009 \@flstop
2010 {\global \colnum \z@}%
2011 }

```

Maybe this should be a loop over #1?

```

2012 \def \@flstop [#1]{%
2013 \if t#1%
2014 \global \topnum \z@
2015 \fi
2016 \if b#1%
2017 \global \botnum \z@
2018 \fi
2019 }

```

*(End definition for `\suppressfloats` and `\@flstop`.)*

Manipulation of float placement and type; both their strings and the corresponding count registers.

**\@fpstype** First a new count register to go with **\@currtype**.  
**\@reqcolroom** Then a new skip register, for information needed to remove the **\@maxsep** conservatism: it is possible that this could use a temporary register.  
**\@textfloatsheight** Finally a dimension register to hold the total height of in-text floats on the current page. This is needed to implement a major change in the functionality of **\@addtocurcol** which is, nevertheless, a bug fix. It is not local and therefore cannot be a temporary register.

```

2020 \newcount \@fpstype
2021 \newdimen \@reqcolroom
2022 \newdimen \@textfloatsheight
2023 {/2ekernel}

```

(End definition for **\@fpstype**, **\@reqcolroom**, and **\@textfloatsheight**.)

**\@fpsadddefault** Adds the default placement to what is already there.  
 Should not need to change this, but could do it as follows:

```

def \@fpsadddefault {%
 \@temptokena \expandafter\expandafter\expandafter
 {\csname fps@\@capttype \endcsname}%
 \edef \reserved@a {\the\@temptokena}%
 \onelevel@sanitize \reserved@a
 \edef \@fps {\@fps\reserved@a}%

```

```

2024 {*2ekernel | fltrace}
2025 \def \@fpsadddefault {%
2026 (*trace)
2027 \fl@trace{fps changed from: \@fps}%
2028 }/trace
2029 \edef \@fps {\@fps\csname fps@\@capttype \endcsname}%
2030 \@latex@warning {%
2031 No positions in optional float specifier.\MessageBreak
2032 Default added (so using '\@fps')}%
2033 }

```

(End definition for **\@fpsadddefault**.)

**\@setfloattypecounts** Sets counters **\@fpstype** and **\@currtype**.  
 BANG == bit4 of **\count\@currbox** = 0.

```

2034 \def \@setfloattypecounts {%
2035 \@currtype \count\@currbox
2036 \@fpstype \count\@currbox
2037 \divide\@currtype\@xxxii \multiply\@currtype\@xxxii
2038 \advance \@fpstype -\@currtype
2039 }(*trace)
2040 \fl@trace{(mod 32) fpstype: \the \@fpstype}%
2041 \fl@trace{(mult of 32) currtype: \the \@currtype}%
2042 % Tracing only: but some should be changed into real errors/warnings?
2043 \ifnum \@fpstype<\sixt@n
2044 \ifnum \@fpstype=\z@
2045 \fl@trace{ERROR: no PLACEMENT, fpstype = \the \@fpstype = 0?}%
2046 \fi

```

```

2047 \ifnum \@fpstype=\@ne
2048 \f@trace{WARNING: only h, fpstype = \the \@fpstype = 1?}%
2049 \fi
2050 \f@trace{BANG float}%
2051 \else
2052 \ifnum \@fpstype=\sixt@n
2053 \f@trace{ERROR: no PLACEMENT, fpstype = \the \@fpstype = 16?}%
2054 \fi
2055 \ifnum \@fpstype=17
2056 \f@trace{WARNING: only h, fpstype = \the \@fpstype = 17?}%
2057 \fi
2058 \f@trace{ORD float}%
2059 \fi
2060 (/trace)
2061 }
2062 (/2ekernel | fltrace)

```

*(End definition for \@setfloattypecounts.)*

Macros for getting, testing and setting bits of the fps.

\@getfpsbit Sets \@tempcnta to required bit of \count\@currbox.

```

2063 (*2ekernel)
2064 \def \@getfpsbit {%
2065 \@boxfpsbit \@currbox
2066 }

```

*(End definition for \@getfpsbit.)*

\@boxfpsbit Used above.

```

2067 \def \@boxfpsbit #1#2{%
2068 \@tempcnta \count#1%
2069 \divide \@tempcnta #2\relax
2070 }

```

*(End definition for \@boxfpsbit.)*

\@testfp New definition of the float page test.

```

2071 \def \@testfp #1{%
2072 \@boxfpsbit #18\relax % Really '#1 8' for human readers!
2073 \ifodd \@tempcnta
2074 \else
2075 \@testtrue
2076 \fi
2077 }

```

*(End definition for \@testfp.)*

\@setfpsbit Sets required bit of \@tempcnta (to 1).

```

2078 \def \@setfpsbit #1{%
2079 \@tempcntb \@tempcnta
2080 \divide \@tempcntb #1\relax
2081 \ifodd \@tempcntb
2082 \else
2083 \advance \@tempcnta #1\relax

```

```

2084 \fi
2085 }
2086 </2ekernel>

```

(End definition for `\@setfpsbit`.)

`\@resethfps` Globally adds t as a possible location for an h or !h only placement: this must be done using the count.

Although it will leave `\@fpstype` set to 17 even if it was originally 1, this does not matter since it is the last thing in `\@addtocurcol`.

```

2087 <*2ekernel | ftrace>
2088 \def \@resethfps {%
2089 \let\reserved@a\empty
2090 \ifnum \@fpstype=\@ne
2091 \def \reserved@a {!}%
2092 \@fpstype 17
2093 \fi
2094 \ifnum \@fpstype=17
2095 \global \advance \count\@currbox \tw@
2096 \@latex@warning@no@line {%
2097 ‘\reserved@a h’ float specifier changed to ‘\reserved@a ht’}%
2098 <*trace>
2099 \f@trace{%
2100 ‘t’ added to ‘\reserved@a h’- new Count: \the \count\@currbox}%
2101 </trace>
2102 \fi
2103 }

```

(End definition for `\@resethfps`.)

Special stuff for BANG floats.

`\@flsetnum` Ignores any zero float counter value in case BANG.

It uses a local assignment to the normally global counter: a bit naughty, perhaps?

These assignments are safe so long as the counter involved is only consulted once (i.e. only for the ‘bang float’) with the changed value. This is the case within `\@addtocurcol` because it is used only once within a call of the output routine (which forms a group).

For `\@addtonextcol` this is achieved by putting a group around its code; this is needed because it is called (by `\@startcolumn`) for each float which was on the deferlist. Almost identical considerations pertain to `\@addtobblcol`. There may be more efficient ways to handle this, but the group seems to be the simplest.

```

2104 \def \@flsetnum #1{%
2105 <*trace>
2106 \f@trace{fpstype: \the \@fpstype (flsetnum \string#1)}%
2107 </trace>
2108 \ifnum \@fpstype<\sixt@n
2109 \ifnum #1=\z@
2110 <*trace>
2111 \f@trace{BANG float resetting \string#1 to 1}%
2112 </trace>
2113 #1\@ne
2114 \fi
2115 \fi

```

```

2116 {*trace}
2117 \fl@trace{\#1 (before) = \the #1}%
2118
```

(End definition for \@flsetnum.)

\@flsettextmin This ignores \textfraction space restriction in case BANG.

```

2120 \def \@flsettextmin {%
2121 {*trace}
2122 \fl@trace{fpstype: \the \@fpstype (flsettextmin)}%
2123
```

```

2124 \ifnum \@fpstype<\sixt@n
2125
```

```

2126 \fl@trace{BANG ignoring textmin}%
2127
```

```

2128 \textmin \z@
2129
```

```

2130
```

```

2131
```

```

2132
```

```

2133
```

```

2134
```

```

2135 }

```

(End definition for \@flsettextmin.)

\@flcheckspace This ignores space restriction in case BANG; this is still slightly conservative since it does not allow for the fact that, if there is no text in the column then \textfloatsep is not needed. Sets @tempswa true if there is room for \@currbox.

```

2136 \def \@flcheckspace #1#2{%
2137 \advance \@reqcolroom
2138 \ifx #2\empty \textfloatsep \else \floatsep \fi
2139
```

```

2140
```

```

2141 \fl@trace{colroom = \the \@colroom
2142 (flcheckspace \string#1 \string#2)}%
2143 \fl@trace{reqcolroom = \the \@reqcolroom
2144 (flcheckspace \string#1 \string#2)}%
2145
```

```

2146 \ifdim \@colroom>\@reqcolroom
2147 \ifdim #1>\ht\@currbox
2148 \tempswatru
2149
```

```

2150 \fl@trace{Space OK: #1 = \the #1 > \the \ht \@currbox
2151 (flcheckspace \string#1 \string#2)}%
2152
```

```

2153
```

```

2154 \fl@trace{fpstype: \the \@fpstype
2155 (flcheckspace \string#1 \string#2)}%
2156
```

```

2157 \ifnum \@fpstype<\sixt@n
2158
```

```

2159 \fl@trace{BANG float ignoring #1

```

```

2160 (flcheckspace \string#1 \string#2):}%
2161 \fl@trace{\@spaces #1 = \the #1. Ht float: \the \ht \currbox
2162 BANG}%
2163
```

`</trace>`

```

2164
```

`\@tempswattrue`

```

2165
```

`<*trace>`

```

2166
```

`\else`

```

2167
```

`\fl@trace{Fail---no room (flcheckspace \string#1 \string#2)
2168 (fpstype \the \@fpstype=ORD?)}%`

```

2169
```

`\fl@trace{\@spaces #1 = \the #1. Ht float: \the \ht \currbox
2170 ORD?}%
2171 
</trace>

```

2172
```

\fi

```

2173
```

\fi

```

2174
```

<*trace>

```

2175
```

\else

```

2176
```

\fl@trace{Fail---no room at 2nd test of colroom
2177 (flcheckspace \string#1 \string#2)}%
2178 
</trace>

```

2179
```

\fi

```

2180 }
```

</2ekernel | fltrace>`

(End definition for `\@flcheckspace`.)

`\@flupdates` This updates everything when a float is placed.

```

2182
```

`<*2ekernel>`

```

2183
```

`\def \@flupdates #1#2#3{%
2184` 
`\global \advance #1\m@ne`

```

2185
\global \advance \@colnum \m@ne

```

2186     
\@tempdima -\ht\currbox

```

2187
\advance \@tempdima

```

2188     
-\ifx #3\empty \textfloatsep \else \floatsep \fi

```

2189
\global \advance #2\@tempdima

```

2190     
\global \advance \@colroom \@tempdima

```

2191
\@cons #3\currbox
2192 }
```

</2ekernel>

```


```


```


```


```


```

(End definition for `\@flupdates`.)

Interesting facts about float mechanisms past and present, together with a summary of various features, some unresolved:

1. The value `\textfraction` does not affect the processing of doublecol floats: this seems sensible, but should be documented.
2. `\twocolumn` floatplacement was wrong: dbl not needed, ord needed.
3. `\@floatplacement` was not called after `\startdblcol` or `\topnewpage`. This has been changed; it is clearly a bug fix.
4. The use `\topnewpage` when `\dblfigrule` is non-trivial produced a rule in the wrong place. This has been fixed by not using `\dblfigrule` when processing the ‘float’ from `\topnewpage`.

5. If the specifier was just h and the float could not be put here, it went on the deferlist and stayed there until a clearpage. It now gets changed to a ‘th’: this is only an error-recovery action, putting just h or !h should be deprecated.
6. `\@dblmaxsep` was ‘the maximum of `\dblfloatsep` and `\dbltexfloatsep`’. But it was never used! Now gone completely, like `\@maxsep`.
7. After an h float is put on a page, it was counted as text when applying the `\textfraction` test; this is possibly too big a change although it is a bug fix?
8. Two consecutive h floats are separated by twice `\intextsep`: this could be changed to one by use of `\addvspace`, OK? Note that it would also mean that less space is put in if an h float immediately follows other spaces. This is also possibly too big a change, at least for compatibility mode? Or it may be simply wrong! It has not been changed.
9. Now `\@addtocurcol` checks first for just p fps. I think that this is an increase in efficiency, but maybe the coding should be made even more efficient.
10. `\@tryfcolumn` now tests if the list is empty first, otherwise lots of wasted time! Thus this test has been removed from `\@startcolumn`. As Frank pointed out, this makes `\@startcolumn` less efficient. But it is now the same as `\@startdblcolumn`: I can see no reason why they should be different, but which is best?
11. Why is `\@colroom` set in `\@doclearpage`?
12. Footnotes. Check what `\clearpage` does when footnotes are left over. Footnotes are not put on float pages and, also, `\@addtonextcol` ignores the existence of held-over footnotes in deciding what floats can go on the page. Not changed.
13. `\clearpage` can still lose non-boxes, at least when floats are involved. It also moves some to the ‘wrong page’, but this may be a coding problem.
14. The ! option makes it necessary to check in `\output` that there is enough room left on the page after adding a float. (This would have been necessary anyway if anyone set `\textmin` too close to zero! A similar danger existed also if the text in a `\twocolumn[text]` entity gets too large.) The current implementation of this also makes the normal case a little less efficient, OK? Not enough room means, at present, less than `\baselineskip`, with a warning: is this OK? Should it be made generic (another parameter)?
15. There are four possibilities for supporting this:  
`\twocolumn[\maketitle more text]`  
One is to change `\maketitle` slightly to allow this. Another is to change `\@topnewpage` so that more than one `\twocolumn[]` command is allowed; in this case `\maketitle\twocolumn[more text]` will work. The former is more robust from the user’s viewpoint, but makes the code for `\maketitle` rather ad hoc (maybe it is already?). Another is to misuse the global `twocolumn` flag locally within `\@topnewpage`. Yet another is to move the column count register from the `multicol` package into the kernel. This has been done.
16. Where should the reinserts be put to maximise the probability that footnotes come out on the correct page? Or should we go for as much compatibility as possible (but see next item)?

17. Should we continue to support (as much as possible) `\samepage`? Some of its intended functionality is now advertised as being provided by `\enlargethispage`. Use of either is likely to result in wrongly placed footnotes, marginalia, etc. Which should have priority: obeying the pagination instructions, or correct placement of notes/marginalia?
18. Is the adjustment of space to cause shrinking in the kludge-\* case correct? Should it be limited to 0pt?
19. Is the setting of `\boxmaxdepth` in `makecol` and friends needed? It only has any effect if `\@textbottom` ends with a box or rule, in which case the vskip to allow for its depth should also be added. If it is kept, it should probably be the last thing in the box. It has now been removed.  
It would perhaps be better to document that `\@textbottom` and `\@texttop` must have natural height 0pt.
20. I cannot see why the vskip adjustment for the depth is needed if `boxmaxdepth` is used to ensure that there is never a too deep box.
21. The value of `\boxmaxdepth` should be explicitly set whenever necessary: it is too risky to assume that it has any particular value. Care is needed in deciding what to set it to.  
It is interesting to note that the value of `\boxmaxdepth` is unique in being read before the local settings for the box group are reset; all other parameter settings which affect the box construction use their values outside the box group.
22. Should `\@maxdepth` store the setting of `\maxdepth` from `lplain`? Or should we provide a proper interface to class files for setting these?

An analysis of various other macros.

`\@opcol` should do `\@floatplacement`, but where? Right at the end, since it always occurs at the start of a column.

```
\def\@opcol{%
 % Why is this done first?
 \global \c@mparbottom \z@
 \if@twocolumn
 \c@outputdblcol
 \else
 \c@outputpage
 % This is not needed since it is done at the end of
 % |\c@outputpage|:
 \global \c@colht \textheight
 \fi}
```

Only tracing has been added to these.

```
2194 <latexrelease | fltrace>\IncludeInRelease{2017/01/01}%
2195 <latexrelease | fltrace> {\c@makefcolumn}{negative height floats}%
2196 <*2ekernel | fltrace | latexrelease>
2197 \def\c@makefcolumn #1{%
2198 \begingroup
```

```

2199 \c@fmin -\maxdimen
2200 \let \c@testfp \c@gobble
2201 \c@tryfcolumn #1%
2202 \endgroup
2203 {*trace}
2204 \if@cfcollmade
2205 \f@l@trace{PAGE: in \string\clearpage
2206 \if@twocolumn ---twocolumn\fi---}%
2207 \f@l@trace{----- float column/page completed from \string#1}%
2208 \fi
2209
```

```

2210 }
2211 \l@texrelease | \f@l@trace \EndIncludeInRelease
2212 \l@texrelease | \f@l@trace \IncludeInRelease{0000/00/00}%
2213 \l@texrelease | \f@l@trace \{\c@makefcolumn}{negative height floats}%
2214 \l@texrelease | \f@l@trace \def \c@makefcolumn #1{%
2215 \l@texrelease | \f@l@trace \begingroup
2216 \l@texrelease | \f@l@trace \c@fmin \z@
2217 \l@texrelease | \f@l@trace \let \c@testfp \c@gobble
2218 \l@texrelease | \f@l@trace \c@tryfcolumn #1%
2219 \l@texrelease | \f@l@trace \endgroup
2220 {*trace}
2221 \l@texrelease | \f@l@trace \if@cfcollmade
2222 \l@texrelease | \f@l@trace \f@l@trace{PAGE: in \string\clearpage
2223 \l@texrelease | \f@l@trace \if@twocolumn ---twocolumn\fi---}%
2224 \l@texrelease | \f@l@trace \f@l@trace{----- float column/page completed
2225 \l@texrelease | \f@l@trace from \string#1}%
2226 \l@texrelease | \f@l@trace \fi
2227
```

```

2228 \l@texrelease | \f@l@trace \}
2229 \l@texrelease | \f@l@trace \EndIncludeInRelease
230
```

This will line up the last baselines in the two columns provided they are constructed in the normal way: i.e. ending in a skip of minus the original depth, with `\c@textbottom` adding nothing.

Thus again it is essential for `\c@textbottom` to have depth 0pt.

```

2231 \l@texrelease | \f@l@trace \IncludeInRelease{2015/01/01}%
2232 \l@texrelease | \f@l@trace \{\c@outputdblcol}{2 column marks}%
2233
```

This is just a change to the single command `\c@outputdblcol` so that it saves mark information for the first column and restores it in the second column.

```

2234 \def \c@outputdblcol{%
2235 \if@cfirstcolumn
2236 \global \c@firstcolumnfalse

```

Save the left column

```

2237 \global \setbox \c@leftcolumn \copy \c@outputbox
2238
```

Remember the marks from the first column

```

2239 \splitmaxdepth \maxdimen
2240 \vbadness \maxdimen

```

In case of `\enlargethispage` we will have infinite negative glue at the bottom of the page (coming from `\vss`) and that will earn us an error message if we `\vsplit` to get at the marks. So we need to remove the last glue (if any) at the end of `\@outputbox` as we are only interested in marks that change doesn't matter.

```
2241 \setbox\@outputbox\vbox{\unvbox\@outputbox\unskip}%
2242 \setbox\@outputbox\vsplit\@outputbox to\maxdimen
```

One minor difference from the current `fixmarks` package, pass the marks through a token register to stop any # tokens causing an error in a `\def`.

```
2243 \toks@\expandafter{\topmark}%
2244 \xdef\@firstcoltopmark{\the\toks@}%
2245 \toks@\expandafter{\splitfirstmark}%
2246 \xdef\@firstcolfirstmark{\the\toks@}%
```

This test does not work if truly empty marks have been inserted, but L<sup>A</sup>T<sub>E</sub>X marks should always have (at least) two brace groups. (Except before the first mark is used, when the marks are empty, but that is OK here.)

```
2247 \ifx\@firstcolfirstmark\empty
2248 \global\let\@setmarks\relax
2249 \else
2250 \gdef\@setmarks{%
2251 \let\firstmark\@firstcolfirstmark
2252 \let\topmark\@firstcoltopmark}%
2253 \fi
2254 End of change
2255 \else
2256 \global\@firstcolumntrue
2257 \setbox\@outputbox\vbox{%
2258 \hb@xt@\textwidth{%
2259 \hb@xt@\columnwidth{\box\@leftcolumn \hss}}%
2260 \hfil
2261 }%
2262 \hb@xt@\columnwidth{\box\@outputbox \hss}}}%
2263 <ftrace> \fl@trace{PAGE: second column also boxed}%
2264 \@combinedblfloats
```

The color of the `\vrule` should be `\normalcolor` as to not inherit the color from the column.

```
2260 {\normalcolor\vrule \@width\columnseprule}%
2261 \hfil
2262 \hb@xt@\columnwidth{\box\@outputbox \hss}}}%
2263 <ftrace> \fl@trace{PAGE: second column also boxed}%
2264 \@combinedblfloats
```

Override current first and top with those of first column if necessary

```
2265 \@setmarks
2266 End of change
2267 \@outputpage
2268 <ftrace> \fl@trace{PAGE: two column page completed}%
2269 \begingroup
2270 \@dblfloatplacement
2271 \@startdblcolumn
2272 \@whilesw\if@fcollmade \fi{\@outputpage
2273 <ftrace> \fl@trace{PAGE: double float page completed}%
2274 \@startdblcolumn}%
2275 \endgroup
2276 \fi}%
```

```

2276 <{latexrelease | fltrace}>\EndIncludeInRelease
2277 <{latexrelease | fltrace}>\IncludeInRelease{0000/00/00}%
2278 <{latexrelease | fltrace}> {\@outputdblcol}{2 column marks}%
2279 <{latexrelease | fltrace}> \def\@outputdblcol{%
2280 <{latexrelease | fltrace}> \if@firstcolumn
2281 <{latexrelease | fltrace}> \global \if@firstcolumnfalse
2282 <{latexrelease | fltrace}> \global \setbox\@leftcolumn \box\@outputbox
2283 <{*trace}>
2284 <{latexrelease | fltrace}> \f1@trace{PAGE: first column boxed}%
2285 </trace>
2286 <{latexrelease | fltrace}> \else
2287 <{latexrelease | fltrace}> \global \if@firstcolumntrue
2288 <{latexrelease | fltrace}> \setbox\@outputbox \vbox {%
2289 <{latexrelease | fltrace}> \hb@xt@\textwidth {%
2290 <{latexrelease | fltrace}> \hb@xt@\columnwidth {%
2291 <{latexrelease | fltrace}> \box\@leftcolumn \hss}%
2292 <{latexrelease | fltrace}> \hfil
2293 <{latexrelease | fltrace}> {\normalcolor\vrule
2294 <{latexrelease | fltrace}> \@width\columnseprule}%
2295 <{latexrelease | fltrace}> \hfil
2296 <{latexrelease | fltrace}> \hb@xt@\columnwidth {%
2297 <{latexrelease | fltrace}> \box\@outputbox \hss}%
2298 <{latexrelease | fltrace}> }%
2299 <{latexrelease | fltrace}> }%
2300 <{*trace}>
2301 <{latexrelease | fltrace}> \f1@trace{PAGE: second column also boxed}%
2302 </trace>
2303 <{latexrelease | fltrace}> \@combinedblfloats
2304 <{latexrelease | fltrace}> \@outputpage
2305 <{*trace}>
2306 <{latexrelease | fltrace}> \f1@trace{PAGE: two column page completed}%
2307 </trace>
2308 <{latexrelease | fltrace}> \begingroup
2309 <{latexrelease | fltrace}> \@dblfloatplacement
2310 <{latexrelease | fltrace}> \@startdblcolumn

```

This loop could be replaced by an `\expandafter` tail recursion in `\@startdblcolumn`.

```

2311 <{latexrelease | fltrace}> \@whilesw\if@fcolmade \fi
2312 <{latexrelease | fltrace}> {\@outputpage
2313 <{*trace}>
2314 <{latexrelease | fltrace}> \f1@trace{PAGE: double float page completed}%
2315 </trace>
2316 <{latexrelease | fltrace}> \@startdblcolumn}%
2317 <{latexrelease | fltrace}> \endgroup
2318 <{latexrelease | fltrace}> \fi
2319 <{latexrelease | fltrace}> }%
2320 <{latexrelease | fltrace}>\EndIncludeInRelease
2321 </2ekernel | fltrace | latexrelease>

```

### 1.1.3 Float placement parameters

The main purpose of this section is to ensure that all the float-placement parameters which need to be set in a class file or package have been declared. It also describes their

use and sets values for them which are reasonable for typical documents using US letter or A4 sized paper.

### Limits for the placement of floating objects

- \c@topnumber This counter holds the maximum number of floats that can appear at the top of a text page or column.

```
2322 {*2ekernel}
2323 \newcount\c@topnumber
2324 \setcounter{topnumber}{2}
```

(*End definition for \c@topnumber.*)

- \topfraction This macro holds the maximum proportion (as a decimal number) of a text page or column that can be occupied by floats at the top.

```
2325 \newcommand\topfraction{.7}
```

(*End definition for \topfraction.*)

- \c@bottomnumber This counter holds the maximum number of floats that can appear at the bottom of a text page or column.

```
2326 \newcount\c@bottomnumber
2327 \setcounter{bottomnumber}{1}
```

(*End definition for \c@bottomnumber.*)

- \bottomfraction This macro holds the maximum proportion (as a decimal number) of a text page or column that can be occupied by floats at the bottom.

```
2328 \newcommand\bottomfraction{.3}
```

(*End definition for \bottomfraction.*)

- \c@totalnumber This counter holds the maximum number of floats that can appear on any text page or column.

```
2329 \newcount\c@totalnumber
2330 \setcounter{totalnumber}{3}
```

(*End definition for \c@totalnumber.*)

- \textfraction This macro holds the minimum proportion (as a decimal number) of a text page or column that must be occupied by text.

```
2331 \newcommand\textfraction{.2}
```

(*End definition for \textfraction.*)

- \floatpagefraction This macro holds the minimum proportion (as a decimal number) of a page or column that must be occupied by floating objects before a ‘float page’ is produced.

```
2332 \newcommand\floatpagefraction{.5}
```

(*End definition for \floatpagefraction.*)

- \c@dbltopnumber This counter holds the maximum number of double-column floats that can appear on the top of a two-column text page.

```
2333 \newcount\c@dbltopnumber
2334 \setcounter{dbltopnumber}{2}
```

(End definition for `\c@dbltopnumber`.)

- `\dbltopfraction` This macro holds the maximum proportion (as a decimal number) of a two-column text page that can be occupied by double-column floats at the top.

2335 `\newcommand{\dbltopfraction}{.7}`

(End definition for `\dbltopfraction`.)

- `\dblfloatpagefraction` This macro holds the minimum proportion (as a decimal number) of a page that must be occupied by double-column floating objects before a ‘double-column float page’ is produced.

2336 `\newcommand{\dblfloatpagefraction}{.5}`

(End definition for `\dblfloatpagefraction`.)

### Floats on a text page

- `\floatsep`  
`\textfloatsep`  
`\intextsep` When a floating object is placed on a page with text, these parameters control the separation between the float and the other objects on the page. These parameters are used for both one-column mode and single-column floats in two-column mode. They are all rubber lengths.

`\floatsep` is the space between adjacent floats that are placed at the top or bottom of the text page or column.

`\textfloatsep` is the space between the main text and floats at the top or bottom of the page or column.

`\intextsep` is the space between in-text floats and the text.

2337 `\newskip{\floatsep}`

2338 `\newskip{\textfloatsep}`

2339 `\newskip{\intextsep}`

2340 `\setlength{\floatsep}{12\p@ \oplus 2\p@ \ominus 2\p@}`

2341 `\setlength{\textfloatsep}{20\p@ \oplus 2\p@ \ominus 4\p@}`

2342 `\setlength{\intextsep}{12\p@ \oplus 2\p@ \ominus 2\p@}`

(End definition for `\floatsep`, `\textfloatsep`, and `\intextsep`.)

- `\dblfloatsep`  
`\dbltextfloatsep` When double-column floats (floating objects that span the whole `\textwidth`) are placed at the top of a text page in two-column mode, the separation between the float and the text is controlled by `\dblfloatsep` and `\dbltextfloatsep`. They are rubber lengths.

`\dblfloatsep` is the space between adjacent double-column floats placed at the top of the text page.

`\dbltextfloatsep` is the space between the main text and double-column floats at the top of the page.

2343 `\newskip{\dblfloatsep}`

2344 `\newskip{\dbltextfloatsep}`

2345 `\setlength{\dblfloatsep}{12\p@ \oplus 2\p@ \ominus 2\p@}`

2346 `\setlength{\dbltextfloatsep}{20\p@ \oplus 2\p@ \ominus 4\p@}`

(End definition for `\dblfloatsep` and `\dbltextfloatsep`.)

### Floats on their own page or column

- \@fptop When floating objects are placed on a separate page or column, called a ‘float page’, the layout of the page is controlled by these parameters, which are rubber lengths.  
\@fpsep  
\@fpbot At the top of the page \@fptop is inserted; typically this supplies some stretchable whitespace. At the bottom of the page \@fpbot is inserted. Between adjacent floats \@fpsep is inserted.

These parameters are used for all floating objects on a ‘float page’ in one-column mode, and for single-column floats in two-column mode.

Note that at least one of the two parameters \@fptop and \@fpbot should contain a **plus** ...fil so as to fill the remaining empty space.

```
2347 \newskip\@fptop
2348 \newskip\@fpsep
2349 \newskip\@fpbot
2350 \setlength\@fptop{0\p@ \cplus 1fil}
2351 \setlength\@fpsep{8\p@ \cplus 2fil}
2352 \setlength\@fpbot{0\p@ \cplus 1fil}
```

(End definition for \@fptop, \@fpsep, and \@fpbot.)

- \@dblftop Double-column ‘float pages’ in two-column mode use similar parameters.  
\@dblfpsep  
\@dblfpbot  
2353 \newskip\@dblftop
2354 \newskip\@dblfpsep
2355 \newskip\@dblfpbot
2356 \setlength\@dblftop{0\p@ \cplus 1fil}
2357 \setlength\@dblfpsep{8\p@ \cplus 2fil}
2358 \setlength\@dblfpbot{0\p@ \cplus 1fil}

(End definition for \@dblftop, \@dblfpsep, and \@dblfpbot.)

- \topfigrule The macros can be used to put in rules between floats and text; whatever they insert  
\botfigrule should be vertical mode material which takes up zero space.  
\dblfigrule  
2359 \let\topfigrule=\relax
2360 \let\botfigrule=\relax
2361 \let\dblfigrule=\relax
2362

(End definition for \topfigrule, \botfigrule, and \dblfigrule.)

## File Y

### lthyphen.dtx

This file contains the code for loading hyphenation patterns into L<sup>A</sup>T<sub>E</sub>X. Most of this will end up in a file called `hyphen.ltx`. If you wish to customize your L<sup>A</sup>T<sub>E</sub>X system in respect of hyphenation patterns, write a file `hyphen.cfg`. If this file exists, it will be loaded instead of `hyphen.ltx`. See the comments below for additional information.

To produce the printed version of this file the following code is used. It can be extracted with the `DOCSTRIP` program, or one can run this file directly through L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub> .

```
1 <*driver>
2 \documentclass{ltxdoc}
3 \begin{document}
4 \DocInput{lthyphen.dtx}
5 \end{document}
6 </driver>
```

The default file `hyphen.ltx` loads hyphenation patterns for US english. If you want to load additional or other hyphenation patterns, you should create a file `hyphen.cfg`. This is best done by starting from `hyphen.ltx`.

For backward compatibility, the default file, `hyphen.ltx`, first tries to load the file `hyphen.tex`. If this file exists, an information message is issued and the appropriate defaults for T<sub>E</sub>X's internal parameters are set: `\language` is initialized to 0, and `\lefthyphenmin` and `\righthyphenmin` to 2 and 3, respectively, to disallow x- or -xx breaks.

```
7 <*default>
8 \InputIfFileExists{hyphen.tex}%
9 {\message{Loading hyphenation patterns for US english.}%
10 \language=0
11 \lefthyphenmin=2 \righthyphenmin=3 }%
```

Otherwise, since we cannot do anything without any hyphenation patterns, an error message is printed and the IniT<sub>E</sub>X run is terminated by invoking `\@@end` (which is the L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub>  name for T<sub>E</sub>X's `\end` primitive).

```
12 {\errhelp{The configuration for hyphenation is incorrectly
13 installed.^^J%
14 If you don't understand this error message you need
15 to seek^Jexpert advice.}%
16 \errmessage{OOPS! I can't find any hyphenation patterns for
17 US english.^^J \space Think of getting some or the
18 latex2e setup will never succeed}\@@end}
19 </default>
```

The following example describes the possible contents of a file `hyphen.cfg` that will load both US English and German hyphenation patterns, making the former the default. It sets `\language` to 0 for the US patterns and to 1 for the German patterns. Then `\language` is set to 0 to make this the default and the default values of `\lefthyphenmin` and `\righthyphenmin` are set.

```
language=0
input hyphen % (or \input ushyphen1 if the file has been renamed)
language=1
input ghyp31
```

```
language=0
lefthyphenmin=2
righthyphenmin=3
endinput
```

Another possibility is to use the package **babel**, by Johannes Braams. That package is distributed with a suitable **hyphen.cfg** file.

# File Z

## ltfinal.dtx

### 1 Final settings

This section contains the final settings for L<sup>A</sup>T<sub>E</sub>X. It initializes some debugging and typesetting parameters, sets the default \catcodes and uc/lc codes, and inputs the hyphenation file.

#### 1.1 Debugging

By default, L<sup>A</sup>T<sub>E</sub>X shows statistics:

```
1 {*2ekernel}
2 \tracingstats1
```

#### 1.2 Typesetting parameters

\@lowpenalty These are penalties used internally.  
\@medpenalty  
\@highpenalty

```
3 \newcount\@lowpenalty
4 \newcount\@medpenalty
5 \newcount\@highpenalty
```

(End definition for \@lowpenalty, \@medpenalty, and \@highpenalty.)

\newmarks Allocate extended marks types if etex is active. Placed here at the end of the format to increase compatibility with count allocations in earlier releases.

```
6 {/2ekernel}
7 {*2ekernel | latexrelease}
8 {latexrelease}\IncludeInRelease{2015/01/01}%
9 {latexrelease} {\newmarks}{Extended Allocation}%
10 \ifx\marks\@undefined\else
11 \def\newmarks{%
12 \e@alloc\marks \e@alloc@chardef{\count256}\m@ne\@alloc@top}
13 \fi
14 {/2ekernel | latexrelease}
15 {latexrelease}\EndIncludeInRelease
16 {latexrelease}\IncludeInRelease{0000/00/00}%
17 {latexrelease} {\newmarks}{Extended Allocation}%
18 {latexrelease}\let\newmarks\@undefined
19 {latexrelease}\EndIncludeInRelease
20 {*2ekernel}
```

(End definition for \newmarks.)

\newXeTeXintercharclass Allocate \XeTeXintercharclass types if xetex is active. previously defined in `xetex.ini`.  
\xe@alloc@intercharclass  
\e@alloc@intercharclass@top

```
21 {/2ekernel}
22 {*2ekernel | latexrelease}
23 {latexrelease}\IncludeInRelease{2015/01/01}%
24 {latexrelease} {\newXeTeXintercharclass}{Extended Allocation}%
```

Classes allocated 1 to 4094 (or 254 on older xetex) (In earlier XeLaTeX versions 1, 2 and 3 were pre-set for CJK).

```

25 \ifx\XeTeXcharclass\@undefined
26 \else
27 \ifdim\the\XeTeXversion\XeTeXrevision\p@>0.99993\p@
28 \chardef\@alloc@intercharclass@top=4095
29 \else
30 \chardef\@alloc@intercharclass@top=255
31 \fi
32 \def\newXeTeXintercharclass{%
33 \e@alloc\XeTeXcharclass
34 \chardef\xe@alloc@intercharclass\m@ne\@alloc@intercharclass@top}
35 \fi
36 </2ekernel | latexrelease>
37 <latexrelease>\EndIncludeInRelease
38 <latexrelease>\IncludeInRelease{0000/00/00}%
39 <latexrelease> {\newXeTeXintercharclass}{Extended Allocation}%
40 <latexrelease> \ifx\XeTeXcharclass\@undefined
41 <latexrelease> \else
42 <latexrelease> \def\xe@alloc@#1#2#3#4#5{\global\advance#1\@ne
43 <latexrelease> \xe@ch@ck#1#4#2%
44 <latexrelease> \allocationnumber#1%
45 <latexrelease> \global#3#5\allocationnumber
46 <latexrelease> \wlog{\string#5=\string#2\the\allocationnumber}}
47 <latexrelease> \def\xe@ch@ck#1#2#3{%
48 <latexrelease> \ifnum#1<#2\else
49 <latexrelease> \errmessage{No room for a new #3}%
50 <latexrelease> \fi}
51 <latexrelease> \def\newXeTeXintercharclass{%
52 <latexrelease> \xe@alloc@\xe@alloc@intercharclass
53 <latexrelease> \XeTeXcharclass\chardef\@cclv}
54 <latexrelease> \fi
55 <latexrelease>\EndIncludeInRelease
56 <*2ekernel | latexrelease>
57 <latexrelease>\IncludeInRelease{2016/02/01}%
58 <latexrelease> {\xe@alloc@intercharclass}{Start of XeTeX class allocator}%
59 \ifx\XeTeXcharclass\@undefined
60 \else
61 \countdef\xe@alloc@intercharclass=257
62 \xe@alloc@intercharclass=\z@
63 \fi
64 </2ekernel | latexrelease>
65 <latexrelease>\EndIncludeInRelease
66 <latexrelease>\IncludeInRelease{2015/01/01}%
67 <latexrelease> {\xe@alloc@intercharclass}{Start of XeTeX class allocator}%
68 <latexrelease> \ifx\XeTeXcharclass\@undefined
69 <latexrelease> \else
70 <latexrelease> \xe@alloc@intercharclass=\thr@@
71 <latexrelease> \fi
72 <latexrelease>\EndIncludeInRelease
73 <latexrelease>\IncludeInRelease{0000/00/00}%
74 <latexrelease> {\xe@alloc@intercharclass}{Start of XeTeX class allocator}%
75 <latexrelease> \ifx\XeTeXcharclass\@undefined

```

```

76 〈\latexrelease〉 \else
77 〈\latexrelease〉 \newcount\xe@alloc@intercharclass
78 〈\latexrelease〉 \xe@alloc@intercharclass=\thr@@
79 〈\latexrelease〉 \fi
80 〈\latexrelease〉\EndIncludeInRelease
81 〈*2ekernel〉

(End definition for \newXeTeXintercharclass, \xe@alloc@intercharclass, and
 \e@alloc@intercharclass@top.)

```

`trace_stack_levels` Now define the Lua function to emulate `\tracingstacklevels` and install it in the `input_level_string` callback.

```

82 〈/2ekernel〉
83 〈*2ekernel | \latexrelease〉

```

In `\latexrelease` mode we always remove the function from the callback, then add the correct version later.

```

84 〈\latexrelease〉\ifx\directlua\@undefined
85 〈\latexrelease〉\else
86 〈\latexrelease〉 \directlua{%
87 〈\latexrelease〉 if luatexbase.callbacktypes['input_level_string'] and %
88 〈\latexrelease〉 luatexbase.in_callback('input_level_string','tracingstacklevels') then
89 〈\latexrelease〉 luatexbase.remove_from_callback('input_level_string','tracingstacklevels')
90 〈\latexrelease〉 end}%
91 〈\latexrelease〉\fi
92 〈\latexrelease〉\IncludeInRelease{2021/06/01}{trace_stack_levels}%
93 〈\latexrelease〉 {Lua trace_stack_levels function}%
94 〈\ifx\directlua\@undefined
95 〈\else
96 〈*2ekernel〉
97 〈\expanded{%
98 〈\everyjob{\the\everyjob
99 〈\noexpand%\directlua
100 〈/2ekernel〉
101 〈\directlua{%
102 〈\local function trace_stack_levels (input_ptr)
103 〈\local tracingstacklevels = tex.count.tracingstacklevels
104 〈\if tex.tracingmacros > 0 or input_ptr < tracingstacklevels then
105 〈\if tracingstacklevels > 0 then
106 〈\if input_ptr < tracingstacklevels then
107 〈\return "\string\n\string~" .. string.rep(".",input_ptr)
108 〈\else
109 〈\return "\string~\string~"
110 〈\end
111 〈\else
112 〈\return "\string\n"
113 〈\end
114 〈\else
115 〈\return ""
116 〈\end
117 〈\end
118 〈\latexrelease〉 if luatexbase.callbacktypes['input_level_string'] then
119 〈\luatexbase.add_to_callback('input_level_string',
120 〈\trace_stack_levels,'tracingstacklevels')
121 〈\latexrelease〉 end

```

```

122 }%
123 {*2ekernel}
124 }{%
125 /2ekernel}
126 \fi
127 \end{IncludeInRelease}
128 \end{latexrelease}

```

Then for the full rollback, just do nothing, since the function was already taken out of the rollback above.

```

129 \end{latexrelease}\IncludeInRelease{0000/00/00}{trace_stack_levels}%
130 \end{latexrelease} {Lua trace_stack_levels function}%
131 \end{latexrelease} Nothing here
132 \end{IncludeInRelease}
133 /2ekernel | latexrelease)
134 {*}2ekernel}

```

(End definition for `trace_stack_levels`.)

The default values of the picture and `\fbox` parameters:

```

135 \unitlength = 1pt
136 \fboxsep = 3pt
137 \fboxrule = .4pt

```

The saved value of `TEX`'s `\maxdepth`:

```

138 \cmaxdepth = \maxdepth

```

`\vsize` initialized because a `\clearpage` with `\vsize < \topskip` causes trouble. `\@colroom` and `\@colht` also initialized because `\vsize` may be set to them if a `\clearpage` is done before the `\begin{document}`

```

139 \vsize = 1000pt
140 \c@colroom = \vsize
141 \c@colht = \vsize

```

Initialise `\textheight` `\textwidth` and page style, to avoid internal errors if they are not set by the class.

```

142 \textheight=.5\maxdimen
143 \textwidth=\textheight
144 \ps@empty

```

### 1.3 Lccodes for hyphenation

For 7- and 8-bit engines the assumption of T1 encodings is the basis for the hyphenation patterns. That's not the case for the Unicode engines, where the assumption is engine-native working. The common loader system provides access to data from the Unicode Consortium covering not only `\lccode` but also other related data. The `\lccode` part of that at least needs to be loaded before hyphenation is tackled: Xe`TEX` follows the standard `TEX` route of building patterns into the format. `LuaTEX` doesn't require this data be loaded *here* but it does need to be loaded somewhere. Rather than test for the Unicode engines by name, the approach here is to look for the extended math mode handling both provide: any other engine developed in this area will presumably also provide `\Umathcode`.

```

145 \ifnum 0%
146 \ifx\Umathcode\@undefined\else 1\fi
147 \ifx\XeTeXmathcode\@undefined\else 1\fi

```

```

148 >\z@
149 \message{ Unicode character data,}
150 \input{load-unicode-data}
151 </2ekernel>
152 <|latexrelease|>\IncludeInRelease{2016/02/01}%
153 <|latexrelease|> {\XeTeXintercharclasses}{XeTeX character classes}%
154 <|latexrelease|> \ifx\XeTeXinterchartoks\undefined
155 <|latexrelease|> \else
156 <|latexrelease|> \begingroup
157 <|latexrelease|> \chardef\XeTeXcharclassID = 0 %
158 <|latexrelease|> \chardef\XeTeXcharclassOP = 0 %
159 <|latexrelease|> \chardef\XeTeXcharclassCL = 0 %
160 <|latexrelease|> \chardef\XeTeXcharclassEX = 0 %
161 <|latexrelease|> \chardef\XeTeXcharclassIS = 0 %
162 <|latexrelease|> \chardef\XeTeXcharclassNS = 0 %
163 <|latexrelease|> \chardef\XeTeXcharclassCM = 0 %
164 <|latexrelease|> \input{load-unicode-xetex-classes}
165 <|latexrelease|> \endgroup
166 <|latexrelease|> \global\let\xtxHanGlue\undefined
167 <|latexrelease|> \global\let\xtxHanSpace\undefined
168 <|latexrelease|> \global\XeTeXinterchartoks 0 1 = {}
169 <|latexrelease|> \global\XeTeXinterchartoks 0 2 = {}
170 <|latexrelease|> \global\XeTeXinterchartoks 0 3 = {}
171 <|latexrelease|> \global\XeTeXinterchartoks 1 0 = {}
172 <|latexrelease|> \global\XeTeXinterchartoks 2 0 = {}
173 <|latexrelease|> \global\XeTeXinterchartoks 3 0 = {}
174 <|latexrelease|> \global\XeTeXinterchartoks 1 1 = {}
175 <|latexrelease|> \global\XeTeXinterchartoks 1 2 = {}
176 <|latexrelease|> \global\XeTeXinterchartoks 1 3 = {}
177 <|latexrelease|> \global\XeTeXinterchartoks 2 1 = {}
178 <|latexrelease|> \global\XeTeXinterchartoks 2 2 = {}
179 <|latexrelease|> \global\XeTeXinterchartoks 2 3 = {}
180 <|latexrelease|> \global\XeTeXinterchartoks 3 1 = {}
181 <|latexrelease|> \global\XeTeXinterchartoks 3 2 = {}
182 <|latexrelease|> \global\XeTeXinterchartoks 3 3 = {}
183 <|latexrelease|> \fi
184 <|latexrelease|>\EndIncludeInRelease
185 <|latexrelease|>\IncludeInRelease{0000/00/00}%
186 <|latexrelease|> {\XeTeXintercharclasses}{XeTeX character classes}%
187 <|latexrelease|> \ifx\XeTeXinterchartoks\undefined
188 <|latexrelease|> \else
189 <|latexrelease|> \input{load-unicode-xetex-classes}
190 <|latexrelease|> \gdef\xtxHanGlue{\hskip0pt plus 0.1em\relax}
191 <|latexrelease|> \gdef\xtxHanSpace{\hskip0.2em plus 0.2em minus 0.1em\relax}
192 <|latexrelease|> \global\XeTeXinterchartoks 0 1 = {\xtxHanSpace}
193 <|latexrelease|> \global\XeTeXinterchartoks 0 2 = {\xtxHanSpace}
194 <|latexrelease|> \global\XeTeXinterchartoks 0 3 = {\nobreak\xtxHanSpace}
195 <|latexrelease|> \global\XeTeXinterchartoks 1 0 = {\xtxHanSpace}
196 <|latexrelease|> \global\XeTeXinterchartoks 2 0 = {\nobreak\xtxHanSpace}
197 <|latexrelease|> \global\XeTeXinterchartoks 3 0 = {\xtxHanSpace}
198 <|latexrelease|> \global\XeTeXinterchartoks 1 1 = {\xtxHanGlue}
199 <|latexrelease|> \global\XeTeXinterchartoks 1 2 = {\xtxHanGlue}
200 <|latexrelease|> \global\XeTeXinterchartoks 1 3 = {\nobreak\xtxHanGlue}
201 <|latexrelease|> \global\XeTeXinterchartoks 2 1 = {\nobreak\xtxHanGlue}

```

```

202 <latexrelease> \global\XeTeXinterchartoks 2 2 = {\nobreak\xtxHanGlue}
203 <latexrelease> \global\XeTeXinterchartoks 2 3 = {\xtxHanGlue}
204 <latexrelease> \global\XeTeXinterchartoks 3 1 = {\xtxHanGlue}
205 <latexrelease> \global\XeTeXinterchartoks 3 2 = {\xtxHanGlue}
206 <latexrelease> \global\XeTeXinterchartoks 3 3 = {\nobreak\xtxHanGlue}
207 <latexrelease> \fi
208 <latexrelease>\EndIncludeInRelease
209 <*2ekernel>

```

There is one over-ride that makes sense here (see below for the same for 8-bit engines): setting the lccode for - to itself.

```
210 \lccode`-='`- % default hyphen char
```

The alternative is that a “traditional” engine is in use.

```
211 \else
```

We set things up so that hyphenation files can assume that the default (T1) lccodes are in use (at present this also sets up the uccodes). We temporarily define \reserved@a to apply \reserved@c to all the numbers in the range of its arguments.

```

212 \def\reserved@a#1#2{%
213 \tempcnta#1\relax
214 \tempcntb#2\relax
215 \reserved@c
216 }
217 \def\reserved@c{%
218 \ifnum\tempcnta>\tempcntb\else
219 \reserved@c\tempcnta
220 \advance\tempcnta\@ne
221 \expandafter\reserved@c
222 \fi
223 }

```

Depending on the TeX version, we might not be allowed to do this for non-ASCII characters.

```

224 \def\reserved@c#1{%
225 \count@=#1\advance\count@ by -"20
226 \uccode#1=\count@
227 \lccode#1=#1
228 }
229 \reserved@c{\a}{\z}
230 \reserved@c{"A0}{ "BC}
231 \reserved@c{"E0}{ "FF}

```

The upper case characters need their \uccode and \lccode values set, and their \sfcode set to 999.

```

232 \def\reserved@c#1{%
233 \count@=#1\advance\count@ by "20
234 \uccode#1=#1
235 \lccode#1=\count@
236 \sfcode#1=999
237 }
238 \reserved@c{\A}{\Z}
239 \reserved@c{"80}{ "9C}
240 \reserved@c{"C0}{ "DF}

```

Well, it would be nice if that were correct, but unfortunately, the Cork encoding contains some odd slots whose uccode or lccode isn't quite what you'd expect.

```

241 \uccode`^^Y='\I % dotless i
242 \lccode`^^Y='^^Y % dotless i
243 \uccode`^^Z='\J % dotless j, ae in OT1
244 \lccode`^^Z='^^Z % dotless j, ae in OT1
245 \lccode`^^9d='\i % dotted I
246 \uccode`^^9d='^^9d % dotted I
247 \lccode`^^9e='\^^9e % d-bar
248 \uccode`^^9e='\^^d0 % d-bar

```

Finally here is one that helps hyphenation in the OT1 encoding.

```
249 \lccode`^^[='\^^[% oe in OT1
```

And we also set the \lccode of \- and \textcompwordmark so that they do not prevent hyphenation in the remainder of the word (as suggested by Lars Helström).

```

250 \lccode`^-='^- % default hyphen char
251 \lccode 127=127 % alternate hyphen char
252 \lccode 23 =23 % textcompwordmark in T1

```

End of the conditional to select either Unicode or T1 encoding defaults.

```
253 \fi
```

At this stage, we can install any last-minute expl3 set-up.

```

254 \@expl@finalise@setup@@
255 \def\@expl@finalise@setup@@{}}

```

This is as good a place as any to active a few XeTeX-specific settings

```

256 \ifx\XeTeXuseglyphmetrics@\undefined
257 \else
258 \XeTeXuseglyphmetrics=1 %
259 \XeTeXdashbreakstate=1 %
260 \fi

```

## 1.4 Hyphenation

The following code will be compiled into the format file. It checks for the existence of `hyphen.cfg` in inputs that file if found. Otherwise it inputs `hyphen.ltx`. Note that these are loaded in *before* the \catcodes are set, so local hyphenation files can use 8-bit input.

We try to load the customized hyphenation description file.

```

261 \InputIfFileExists{hyphen.cfg}
262 \typeout{=====
263 Local configuration file hyphen.cfg used^^J%
264 =====}%
265 \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
266 }
267 {\input{hyphen.ltx}}
268 \let\@addtofilelist\gobble

```

```
\l@nohyphenation
```

```

269 \ifx\l@nohyphenation \undefined
270 \newlanguage\l@nohyphenation
271 \fi

```

*(End definition for \l@nohyphenation.)*

\document@default@language Default document language. -1 acts as language 0, but used as a flag in \document to see if it has been set in the preamble.

```
272 〈/2ekernel〉
273 〈*2ekernel | latexrelease〉
274 〈latexrelease〉\IncludeInRelease{2017/04/15}%">
275 〈latexrelease〉 {〈\document@default@language〉{Save language for hyphenation}%">
276 〈\let\document@default@language\m@ne
277 〈/2ekernel | latexrelease〉
278 〈latexrelease〉\EndIncludeInRelease
279 〈latexrelease〉\IncludeInRelease{0000/00/00}%">
280 〈latexrelease〉 {〈\document@default@language〉{Save language for hyphenation}%">
281 %
282 〈latexrelease〉\let\document@default@language\@undefined
283 〈latexrelease〉\EndIncludeInRelease
284 〈*2ekernel〉
```

(End definition for \document@default@language.)

## 1.5 Font loading

Fonts loaded during the formatting process might already have changed the \font@submax from Opt to something higher. If so, we put out a bold warning.

```
285 \ifdim \font@submax >\z@
286 \font@warning{Size substitutions with differences\MessageBreak
287 up to \font@submax\space have occurred.\MessageBreak
288 \MessageBreak
289 Please check the transcript file
290 carefully\MessageBreak
291 and redo the format generation if necessary!
292 \@gobbletwo}%">
293 \errhelp{Only stopped, to give you time to
294 read the above message.}
295 \errmessage{}
```

We reset the macro. Otherwise every user will get a warning on every job.

```
296 \def\font@submax{Opt}
297 \fi
```

For pdftEX preload and enable automatic glyph to Unicode mapping for more reliable copy and paste support.

```
298 〈/2ekernel〉
299 〈*2ekernel | latexrelease〉
300 〈latexrelease〉\IncludeInRelease{2021/06/01}%">
301 〈latexrelease〉 {〈\pdffgentounicode〉{Preload glyptounicode}%">
302 〈\ifx \pdffgentounicode \@undefined \else
303 〈*2ekernel〉
304 〈\ifnum 0=0%
305 〈\ifdefined\pdftexversion
306 % \pdftexversion<140 does not have \pdffgentounicode, so we only check higher values
307 〈\ifnum \pdftexversion=140 \ifnum\pdftexrevision<22 1\fi\fi
308 〈\fi
309 〈\relax
310 〈/2ekernel〉
311 〈\input glyptounicode
```

```

312 {*2ekernel}
313 \else
314 \begingroup
315 \everyeof{\noexpand}\endlinechar-1
316 \edef\x{\endgroup
317 \everyjob{\the\everyjob@@@input glyptounicode }%
318 }\x
319 \fi
320
```

```

321 \pdfgentounicode=1
322 \fi
323
```

```

324
```

```

324 \EndIncludeInRelease

```

When rolling back we can't unload the glyptounicode mappings, but we can reset `\pdfgentounicode` to ensure that they aren't used.

```

325
```

```

326
```

```

327
```

```

328
```

```

329
```

```

330
```

```

331
```

```

331 {*2ekernel}

```

## 1.6 Input encoding

Starting with the 2018 L<sup>A</sup>T<sub>E</sub>X release default the inputencoding to UTF-8. Unless the format is being used with luatex, xetex, encTeX or mltex.

This is done in a way largely compatible with older releases: `utf8.def` is input just as if

```

\usepackage[utf8]{inputenc}

```

had been used, however rather than input the whole package a minimal core part just enough to support loading the UTF-8 encoding files is defined here.

If a document re-specifies UTF-8 this is silently ignored.

```

332
```

```

333
```

Check that a classic 8-bit tex engine is being used (LaTeX or PDFLaTeX).

```

334
```

```

335
```

Skip this section in Unicode TeX, or if MLTeX and EncTeX are enabled.

```

336
```

```

337
```

```

338
```

```

339
```

```

340
```

```

341
```

```

342
```

```

343
```

```

344
```

```

345
```

```

346
```

```

347 \noexpand\IeC
348 \fi
349 }

 Make characters active for UTF-8 input formats

350 \Qtempcnta=1
351 \loop
352 \catcode\Qtempcnta=13 %
353 \advance\Qtempcnta\@ne %
354 \ifnum\Qtempcnta<32 %
355 \repeat %

356 \catcode0=15 % null
357 \catcode9=10 % tab
358 \catcode10=12 % ctrl J
359 \catcode12=13 % ctrl L
360 \catcode13=5 % newline
361 \Qtempcnta=128
362 \loop
363 \catcode\Qtempcnta=13
364 \advance\Qtempcnta\@ne
365 \ifnum\Qtempcnta<256
366 \repeat

```

**\UseRawInputEncoding** Reset 8 bit characters to catcode 12 so the input encoding matches the “Raw” font encoding. Useful for special behaviours, or for compatibility with older L<sup>A</sup>T<sub>E</sub>X formats.

```

367 \def\UseRawInputEncoding{%
368 \let\inputencodingname\@undefined % revert
369 \let\DeclareFontEncoding@\DeclareFontEncoding@saved % revert
370 \let\DeclareUnicodeCharacter\@undefined % revert
371 \Qtempcnta=1
372 \loop
373 \catcode\Qtempcnta=15 %
374 \advance\Qtempcnta\@ne %
375 \ifnum\Qtempcnta<32 %
376 \repeat %

377 \catcode0=15 % null
378 \catcode9=10 % tab
379 \catcode10=12 % ctrl J
380 \catcode12=13 % ctrl L
381 \catcode13=5 % newline
382 \Qtempcnta=128
383 \loop
384 \catcode\Qtempcnta=12
385 \advance\Qtempcnta\@ne
386 \ifnum\Qtempcnta<256
387 \repeat
388 }

```

(End definition for \UseRawInputEncoding.)

**\DeclareFontEncoding@saved** Saved version of \DeclareFontEncoding@ before utf8.def modifies it for use in \UseRawInputEncoding above.

```

389 \let\DeclareFontEncoding@saved\DeclareFontEncoding@

```

(End definition for \DeclareFontEncoding@saved.)

```
390 \edef\inputencodingname{utf8}%
391 \input{utf8.def}
392 \let\UTFviii@undefined@err@@\UTFviii@undefined@err
393 \let\UTFviii@invalid@err@@\UTFviii@invalid@err
394 \let\UTFviii@two@octets@@\UTFviii@two@octets
395 \let\UTFviii@three@octets@@\UTFviii@three@octets
396 \let\UTFviii@four@octets@@\UTFviii@four@octets
397 {2ekernel}\def\UTFviii@undefined@err#1{\gobble#1}%
398 {2ekernel}\let\UTFviii@invalid@err\string
399 {2ekernel}\let\UTFviii@two@octets\string
400 {2ekernel}\let\UTFviii@three@octets\string
401 {2ekernel}\let\UTFviii@four@octets\string
402 {2ekernel}\everyjob\expandafter{\the\everyjob
403 {2ekernel}\let\UTFviii@undefined@err\UTFviii@undefined@err@@
404 {2ekernel}\let\UTFviii@invalid@err\UTFviii@invalid@err@@
405 {2ekernel}\let\UTFviii@two@octets\UTFviii@two@octets@@
406 {2ekernel}\let\UTFviii@three@octets\UTFviii@three@octets@@
407 {2ekernel}\let\UTFviii@four@octets\UTFviii@four@octets@@
408 {2ekernel}}
409 \let\@inpcnt@test\@undefined
410 \let\space@catcode\@undefined
```

For formats not set up for UTF-8 default, set the C0 controls to catcode 15.

```
411 \else
412 \@tempcnta=0
413 \loop
414 \catcode\@tempcnta=15 %
415 \advance\@tempcnta\@ne %
416 \ifnum\@tempcnta<32 %
417 \repeat %
418 \catcode0=15 % null
419 \catcode9=10 % tab
420 \catcode10=12 % ctrl J
421 \catcode12=13 % ctrl L
422 \catcode13=5 % newline
423 \let\UseRawInputEncoding\relax
```

This ends the skipped code in Unicode engines:

```
424 \fi
425 {/2ekernel | latexrelease}
426 {/latexrelease}\EndIncludeInRelease
427 {/latexrelease}\IncludeInRelease{0000/00/00}%
428 {/latexrelease} {\UTFviii@invalid}{UTF-8 default}%
```

The first block of commands got only introduced in 2019 but we revert all of Unicode support in one go not jump to the intermediate version.

```
429 {/latexrelease} \let\UTFviii@two@octets@combine\@undefined
430 {/latexrelease} \let\UTFviii@three@octets@combine\@undefined
431 {/latexrelease} \let\UTFviii@four@octets@combine\@undefined
432 {/latexrelease} \let\UTFviii@two@octets@string\@undefined
433 {/latexrelease} \let\UTFviii@three@octets@string\@undefined
434 {/latexrelease} \let\UTFviii@four@octets@string\@undefined
435 {/latexrelease} \let\UTFviii@two@octets@noexpand\@undefined
```

```

436 <|latexrelease> \let\UTFviii@three@octets@noexpand\@undefined
437 <|latexrelease> \let\UTFviii@four@octets@noexpand\@undefined
438 <|latexrelease>\@tempcnta=0
439 <|latexrelease>\loop
440 <|latexrelease> \catcode\@tempcnta=15
441 <|latexrelease> \advance\@tempcnta\@ne
442 <|latexrelease>\ifnum\@tempcnta<32
443 <|latexrelease>\repeat %
444 <|latexrelease>\catcode9=10 % tab
445 <|latexrelease>\catcode10=12 % ctrl J
446 <|latexrelease>\catcode12=13 % ctrl L
447 <|latexrelease>\catcode13=5 % newline
448 <|latexrelease>\@tempcnta=128
449 <|latexrelease>\loop
450 <|latexrelease>\catcode\@tempcnta=12
451 <|latexrelease>\advance\@tempcnta\@ne
452 <|latexrelease>\ifnum\@tempcnta<256
453 <|latexrelease>\repeat
454 <|latexrelease>\let\IeC\@undefined
455 <|latexrelease>\def\DeclareFontEncoding@#1#2#3{%
456 <|latexrelease> \expandafter
457 <|latexrelease> \ifx\csname T@#1\endcsname\relax
458 <|latexrelease> \def\cdp@elt{\noexpand\cdp@elt}%
459 <|latexrelease> \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
460 <|latexrelease> \default@family}{\default@series}%
461 <|latexrelease> \default@shape}%
462 <|latexrelease> \expandafter\let\csname#1-cmd\endcsname\@changed@cmd
463 <|latexrelease> \else
464 <|latexrelease> \font@info{Redeclaring font encoding #1}%
465 <|latexrelease> \fi
466 <|latexrelease> \global\@namedef{T@#1}{#2}%
467 <|latexrelease> \global\@namedef{M@#1}{\default@M#3}%
468 <|latexrelease> \xdef\LastDeclaredEncoding{#1}%
469 <|latexrelease> }
470 <|latexrelease> \let\UseRawInputEncoding\@undefined
471 <|latexrelease> \let\DeclareFontEncoding@saved\@undefined
472 <|latexrelease> \let\inputencodingname\@undefined
473 <|latexrelease>\EndIncludeInRelease

474 <|*2ekernel>
475 % \begin{macrocode}
476 %
477 % We temporarily define |\reserved@a| to apply |\reserved@c| to all the
478 % numbers in the range of its arguments.
479 % \begin{macrocode}
480 \def\reserved@a#1#2{%
481 \tempcnta#1\relax
482 \tempcntb#2\relax
483 \reserved@b
484 }
485 \def\reserved@b{%
486 \ifnum\@tempcnta>\@tempcntb\else
487 \reserved@c\@tempcnta
488 \advance\@tempcnta\@ne

```

```

489 \expandafter\reserved@b
490 \fi
491 }

```

Set the special catcodes (although some of these are useless, since an error will have occurred if the catcodes have changed). Note that  $\wedge J$  has catcode ‘other’ for use in warning messages.

```

492 \catcode`\ =10
493 \catcode`\#=6
494 \catcode`\$=3
495 \catcode`\%=14
496 \catcode`\&=4
497 \catcode`\\"=0
498 \catcode`\^=7
499 \catcode`_=8
500 \catcode`\{=1
501 \catcode`\}=2
502 \catcode`\-=13
503 \catcode`\@=11
504 \catcode`\wedge I=10
505 \catcode`\wedge J=12
506 \catcode`\wedge L=13
507 \catcode`\wedge M=5

```

Set the ‘other’ catcodes.

```

508 \def\reserved@c#1{\catcode#1=12\relax}
509 \reserved@c{`}
510 \reserved@c{`}
511 \reserved@a{`}{{`?}}
512 \reserved@c{`}
513 \reserved@c{`}
514 \reserved@c{`}
515 \reserved@c{`}

```

Set the ‘letter’ catcodes.

```

516 \def\reserved@c#1{\catcode#1=11\relax}
517 \reserved@a{`A}{{`Z}}
518 \reserved@a{`a}{{`z}}

```

All the characters in the range 0–31 and 127–255 are illegal, *except* tab ( $\wedge I$ ), nl ( $\wedge J$ ), ff ( $\wedge L$ ) and cr ( $\wedge M$ ).

## 1.7 Lccodes and uccodes

We now again set up the default (T1) uc/lccodes. The lower case characters need their `\uccode` and `\lccode` values set. Some of this is a repeat of the set-up before loading hyphenation files. Depending on the TeX version, we might not be allowed to do this for non-ASCII characters. For the Unicode engines (XeTeX and LuaTeX) there is no need to do any of this: they use hyphenation data which does not alter any of the set up and so this entire block is skipped.

```

519 \ifnum 0%
520 \ifx\Umathcode\@undefined\else 1\fi
521 \ifx\XeTeXmathcode\@undefined\else 1\fi
522 >\z@
523 \else

```

```

524 \def\reserved@c#1{%
525 \count@=#1\advance\count@ by -"20
526 \uccode#1=\count@
527 \lccode#1=#1
528 }
529 \reserved@a{\a}{`z}
530 \reserved@a{"A0}{"BC}
531 \reserved@a{"E0}{"FF}

```

The upper case characters need their \uccode and \lccode values set, and their \sfcode set to 999.

```

532 \def\reserved@c#1{%
533 \count@=#1\advance\count@ by "20
534 \uccode#1=#1
535 \lccode#1=\count@
536 \sfcode#1=999
537 }
538 \reserved@a{'A}{`Z}
539 \reserved@a{"80}{"9C}
540 \reserved@a{"CO}{"DF}

```

Well, it would be nice if that were correct, but unfortunately, the Cork encoding contains some odd slots whose uccode or lccode isn't quite what you'd expect.

```

541 \uccode`^Y='I % dotless i
542 \lccode`^Y='^Y % dotless i
543 \uccode`^Z='J % dotless j, ae in OT1
544 \lccode`^Z='^Z % dotless j, ae in OT1
545 \lccode`^9d='i % dotted I
546 \uccode`^9d='^9d % dotted I
547 \lccode`^9e='^9e % d-bar
548 \uccode`^9e='^d0 % d-bar

```

Finally here is one that helps hyphenation in the OT1 encoding.

```

549 \lccode`^^[='^^[% oe in OT1
550 \fi % End of reset block for 8-bit engines

```

\MakeUppercase  
\MakeUppercase  
\@uclclist

And whilst we're doing things with uc/lc tables, here are two commands to upper- and lower-case a string.  
Note that this implementation is subject to change! At the moment we're not providing any way to extend the list of uc/lc commands, since finding a good interface is difficult. These commands have some nasty features, such as uppercasing mathematics, environment names, labels, etc. A much better long-term solution is to use all-caps fonts, but these aren't generally available.

Two wrappers around the L3 case changing functions. \protected to make them mostly safe as replacements for uppercase and \lowercase.

```

551 \ExplSyntaxOn
552 \protected\def\Expl@text@lowercase@@n{\text_lowercase:n}
553 \protected\def\Expl@text@uppercase@@n{\text_uppercase:n}
554 \ExplSyntaxOff
555 \DeclareRobustCommand{\MakeUppercase}[1]{%
556 \def\i{I}\def\j{J}%
557 \def\reserved@a##1##2{\let##1##2\reserved@a}%
558 \expandafter\reserved@a\@uclclist\reserved@b{\reserved@b\@gobble}%
}

```

Tell UTF-8 processing to process chars even though we are in an `\protected@edef`.

```
559 \protected@edef\reserved@a{\@expl@text@uppercase@@n{#1}}%
560 \reserved@a
561 }
562 \DeclareRobustCommand{\MakeLowercase}[1]{%
563 \def\reserved@a##1##2{\let##2##1\reserved@a}%
564 \expandafter\reserved@a\@ucclist\reserved@b{\reserved@b\@gobble}%
565 \protected@edef\reserved@a{\@expl@text@lowercase@@n{#1}}%
566 \reserved@a
567 }
568 \def\@ucclist{\oe\OE\o\O\ae\AE
569 \dh\DH\dj\DJ\l\L\ng\NG\ss\SS\ij\IJ\th\TH}
```

The above code works, but has the nasty side-effect that if you say something like:

```
\markboth{\MakeUppercase\contentsname}
{\MakeUppercase\contentsname}
```

then the uppercasing is only done to the first letter of the contents name, since the mark expands out to:

```
\mark{\protect\MakeUppercase Table of Contents}
{\protect\MakeUppercase Table of Contents}
```

In order to get round this, we redefine `\MakeUppercase` and `\MakeLowercase` to grab their argument and brace it. This is a very low-level hack, and is *not* recommended practice! This is an instance of a general problem that makes it unsafe to grab arguments unbraced, and probably needs a more general solution. For the moment though, this hack will do:

```
570 \protected@edef\MakeUppercase#1{\MakeUppercase{#1}}
571 \protected@edef\MakeLowercase#1{\MakeLowercase{#1}}
```

(*End definition for `\MakeUppercase` and others.*)

## 1.8 Applying Patch files

Between major releases, small patches will be distributed in files `ltpatch.ltx` which must be added at this point.

Patch file code removed.

```
572 %\IfFileExists{ltpatch.ltx}
573 % {\typeout{=====
574 % Applying patch file ltpatch.ltx^^J%
575 % ======}
576 % \def\fmtversion@topatch{unknown}
577 % \input{ltpatch.ltx}
578 % \ifx\fmtversion\fmtversion@topatch
579 % \ifx\patch@level@\undefined
580 % \typeout{^^J^^J^^J%
581 % !!!!!!!}
582 % !! Patch file 'ltpatch.ltx' not suitable for this^^J%
583 % !! version of LaTeX.^^J^^J%
584 % !! Please check if initex found an old patch file:^^J%
585 % !! --- if so, rename it or delete it, and redo the^^J%
```

```

586 % !! initex run.^~J%
587 %
588 % \batchmode \@@end
589 % \else

```

The code below adds the ‘patch level’ string to the first `\typeout` in the startup banner.

```

590 % \def\fmtversion@topatch{0}%
591 % \ifx\fmtversion@topatch\patch@level\else
592 % \def\reserved@a{\typeout{\#1 patch level \patch@level}\#2\reserved@a{%
593 % \typeout{\#1 patch level \patch@level}\#2}
594 % \everyjob\expandafter\expandafter\expandafter{%
595 % \expandafter\reserved@a\the\everyjob\reserved@a}
596 % \let\reserved@a\relax
597 % \the\everyjob
598 % \fi
599 % \fi
600 % \else
601 % \typeout{^~J^~J^~J%
602 % !!!!!!!!!!!!!!!^~J%
603 % !! Patch file ‘ltpatch.ltx’ (for version <\fmtversion@topatch>)^~J%
604 % !! is not suitable for version <\fmtversion> of LaTeX.^~J^~J%
605 % !! Please check if initex found an old patch file:^~J%
606 % !! --- if so, rename it or delete it, and redo the^~J%
607 % !! initex run.^~J%
608 % !!!!!!!!!!!!!!!^~J%
609 % \batchmode \@@end
610 % \fi
611 % \let\fmtversion@topatch\relax
612 % }{}}

```

## 1.9 Freeing Memory

- `\reserved@a` And just to make sure nobody relies on those definitions of `\reserved@b` and friends.  
`\reserved@b` These macros are reserved for use in the kernel. *Do not use them as general scratch macros.*

```

613 \let\reserved@a@filelist
614 \let\reserved@b=\@undefined
615 \let\reserved@c=\@undefined
616 \let\reserved@d=\@undefined
617 \let\reserved@e=\@undefined
618 \let\reserved@f=\@undefined

```

(End definition for `\reserved@a` and `\reserved@b`.)

```

\toks
619 \toks0{}
620 \toks2{}
621 \toks4{}
622 \toks6{}
623 \toks8{}

```

(End definition for `\toks`.)

- `\errhelp` Empty the error help message, which may have some rubbish:  
`624 \errhelp{}`

(End definition for \errhelp.)

## 1.10 Initialise file list

\@providesfile Initialise for use in the document. During initex a modified version has been used which leaves debugging information for `latexbug.tex`.

```
625 \def\@providesfile#1[#2]{%
626 \wlog{File: #1 #2}%
627 \expandafter\xdef\csname ver@#1\endcsname{#2}%
628 }
```

(End definition for \@providesfile.)

\@filelist Reset \@filelist so files input while making the format are not listed. The list built up

\@addtofilelist so far may take up a lot of memory and so it is moved to \reserved@a where it will be overwritten as soon as almost any L<sup>A</sup>T<sub>E</sub>X command is issued in a class file. However the `latexbug.tex` program will be able to access this information and insert it into a bug report.

```
629 \let\@filelist\@gobble
630 \def\@addtofilelist#1{\xdef\@filelist{\@filelist,#1}}%
```

(End definition for \@filelist and \@addtofilelist.)

## 1.11 Preparation for supporting PDF in backends

At the current point in time, basic support for PDF in backends is not part of L<sup>A</sup>T<sub>E</sub>X core; it is provided by external packages. At some time in the future that work will be placed into the kernel but for now it is separate and has to be explicitly loaded in the document.

In that code there is a command \IfPDFManagementActiveTF which can be used by packages in order to execute different code depending on the whether this basic backend support is loaded.

To make this also work properly when this external package is not loaded at all, we here add this command already in the kernel (with a trivial definition); thus any package can query this loading state in all circumstances. Once this basic PDF backend support gets moved to the kernel, this definition will vanish again from here or, rather, it will be replaced by a real test.

\IfPDFManagementActiveTF So long as the code for the basic backend support for PDF is not loaded, the test that is implicit here will always return the false branch. Once this code is loaded, this definition will get replaced by a real test (as it is then possible that the management code is either activated or not activated).

```
631 \let \IfPDFManagementActiveTF \@secondoftwo
```

(End definition for \IfPDFManagementActiveTF.)

## 1.12 Do some temporary work for pre-release

This is a good place to load code that hasn't yet been integrated into the other files ...

## 1.13 Some last minute initializations ...

Load the first aid set of definitions for external packages that await updates.

```
632 \Qinput{latex2e-first-aid-for-external-files.ltx}
```

## 1.14 Dumping the format

Finally we make @ into a letter, ensure the format will be in the ‘normal’ error mode, and dump everything into the format file.

```
633 \makeatother
634 \errorstopmode
635 \dump
636 </2ekernel>
```

# Change History

1985-11-04 ltmath.dtx LaTeX2.09		\mathversion: Test if version defined added. . . . .	418
General: produce warning message if line extends into margin. Doesn't warn about formula overprinting equation number. . . . .	644		
1989-04-10 ltfssbas.dtx v1.0a			
General: Starting with version numbers! \ifmmode added in \math@group . . . . .	407		
1989-04-10 ltfssbas.dtx v1.0b			
General: \preload@sizes added. . . . .	407		
\wrong@fontshape changed to define substitution font/shape macro. . . . .	407		
1989-04-10 ltfssini.dtx v1.0a			
General: Starting with version numbers \newif for \@tempswa added since this switch is unknown at the time when this file is read in. (latex.tex is loaded later.) \math@famname changed to \math@version. . . . .	515		
1989-04-14 ltfssbas.dtx v1.0c			
General: More documentation added. . . . .	407		
1989-04-15 ltfssini.dtx v1.0b			
General: \mathfontset renamed to \mathversion. . . . .	515		
1989-04-19 ltfssbas.dtx v1.0d			
General: Even more doc. . . . .	407		
1989-04-21 ltfssbas.dtx v1.0e			
General: Documentation is fun! Parameters of \define@mathalphabet changed. . . . .	407		
1989-04-21 ltfssini.dtx v1.0c			
General: Changed to conform to fam.tex. . . . .	515		
1989-04-23 ltfssbas.dtx v1.0f			
General: % in \getanddefinefonts added. . . . .	407		
1989-04-26 ltfssini.dtx v1.0d			
General: \xpt added. . . . .	515		
1989-04-27 ltfssbas.dtx v1.0g			
General: Documentation revised. . . . .	407		
1989-04-27 ltfssini.dtx v1.0e			
General: Definitions of L <sup>A</sup> T <sub>E</sub> X symbols corrected. . . . .	515		
1989-04-29 ltfssbas.dtx v1.0h			
General: Documented problem with \halign, and \noalign . . . . .	407		
		\mathversion: Test if version defined added. . . . .	418
1989-04-29 ltfssbas.dtx v1.0i			
General: Removed the \halign \noalign correction (wasn't bugfree) . . . . .	407		
1989-04-29 ltfssini.dtx v1.0f			
General: Corrections to L <sup>A</sup> T <sub>E</sub> X tabular env. added. . . . .	515		
1989-05-01 ltfssbas.dtx v1.0j			
General: Default for \baselinestretch added. . . . .	407		
1989-05-22 ltfssbas.dtx v1.0k			
General: Lines longer than 72 characters folded. . . . .	407		
1989-05-22 ltfssini.dtx v1.0g			
General: Lines shortened to 72 characters . . . . .	515		
1989-09-14 ltfssbas.dtx v1.0m			
General: Global replacement: \group to \mathgroup . . . . .	407		
\mathversion: Corrected typo: \endcsname to \endcsmame. . . . .	418		
1989-11-07 ltfssini.dtx v1.0i			
General: All family, series, and shape names abbreviated. . . . .	515		
1989-11-08 ltfssbas.dtx v1.0o			
General: First parameter of \define@mathalphabet and \define@mathgroup changed from string to control sequence. . . . .	407		
1989-11-14 ltfssbas.dtx v1.0p			
\math@version: Math version prefix 'mv@' added. . . . .	418		
1989-11-19 ltfssbas.dtx v1.0q			
\define@newfont: Group added. . . . .	420		
\wrong@fontshape: Instead of calling \family\default@family, etc. we directly set \f@family, etc. . . . .	425		
1989-11-22 ltfssbas.dtx v1.0r			
\math@version: \def → \edef for \math@version. . . . .	418		
1989-11-25 ltfssbas.dtx v1.0s			
General: All \edef\font@name changed to \xdef\font@name. Necessary after introduction of \begingroup/\endgroup in v1.0q. . . . .	407		
extra// → + in \extra@def. . . . .	407		

1989-11-26 ltfssbas.dtx v1.0t \select@group: \bgroup/\egroup changed to \begingroup/\endgroup to avoid empty Ord atom on math list. . . . .	427	1990-01-25 ltfssini.dtx v1.1e \nfss@text: Macro added. . . . .	538
1989-12-02 ltfssini.dtx v1.1b General: \rmmath renamed to \mathrm . . . . .	515	1990-01-27 ltfssbas.dtx v1.2d \DeclarePreloadSizes: Font identifier set to \relax. . . . .	413
1989-12-03 ltfssini.dtx v1.1c General: Some internal macros renamed to make them inaccessible. . . . .	515	1990-01-28 ltfssbas.dtx v1.2e \mathgroup: \newfam let to \new@mathgroup. . . . .	407
1989-12-05 ltfssbas.dtx v1.0u \addto@hook: \addto@hook added. . . . .	432	1990-01-28 ltfssbas.dtx v1.2f \define@newfont: Added call to \curr@fontshape macro to allow substitution. . . . .	420
1989-12-05 lfsstrc.dtx v1.0u fam.dtx \every@math@size: Hook \every@size added. . . . .	464	\wrong@fontshape: Warning message slightly changed. . . . .	425
1989-12-13 lfsstrc.dtx v1.0f \use@mathgroup: \expandafter added before final \fi. . . . .	467	1990-01-28 ltfssini.dtx v1.2b \em: Call to \nomath added. . . . .	535
1989-12-16 ltfssbas.dtx v1.1a \select@group: \relax in front added. . . . .	427	1990-02-08 ltfssini.dtx v1.1g General: Protected the commands \fam, \series, \shape, \size, \selectfont, and \mathversion. . . . .	515
Now four arguments. . . . .	427	1990-02-16 ltfssbas.dtx v1.2g General: Support for changes of \baselineskip without changing the size. . . . .	407
Redefinition of alphabet now simpler. . . . .	428	\mathversion: \nomath added. . . . .	418
Usage of '=' macro added. . . . .	428	1990-02-18 lfsstrc.dtx v1.0j \selectfont: Redefine unprotected version \p@selectfont instead of \selectfont. . . . .	459
1989-12-16 lfsstrc.dtx v1.1a \selectfont: Changed order of calls. . . . .	459	1990-03-14 lfsstrc.dtx v1.0k General: Added code for TeX3. . . . .	455
\use@mathgroup: Redefinition of alphabet now simpler. . . . .	466	\extract@font: Added code for TeX3. . . . .	458
Usage of '=' macro added. . . . .	466	1990-03-30 ltfssbas.dtx v1.2h \math@egroup: Changed to have one arg. . . . .	430
1990-01-18 lfsstrc.dtx v1.0h General: \tracingfonts meaning changed. . . . .	455	1990-03-30 lfsstrc.dtx v1.2h \use@mathgroup: Third argument removed (see \math@egroup). . . . .	466
1990-01-20 ltfssbas.dtx v1.2a \math@bgroup: Def. placed in this file. . . . .	430	1990-04-01 ltfssbas.dtx v1.2i General: Code added from tracefn.ttx. . . . .	407
\math@egroup: Def. placed in this file. . . . .	430	Support for TeX3. . . . .	407
\select@group: Def for alph id changed. . . . .	428	1990-04-01 lfsstrc.dtx v1.0l General: Part of code moved to fam.dtx. . . . .	455
1990-01-21 ltfssbas.dtx v1.2b \select@group: Code moved to \use@mathgroup. . . . .	428	\tracingfonts: Check if \tracingfonts already defined. . . . .	456
1990-01-21 lfsstrc.dtx v1.2b \use@mathgroup: Macro added to allow cleaner interface. . . . .	466	1990-04-01 lfsstrc.dtx v1.0o \tracingfonts: Check if \tracingfonts defined removed again. . . . .	456
1990-01-23 ltfssbas.dtx v1.2c General: \no@version@warning renamed to \no@alphabet@error. . . . .	407		
Macro \no@alphabet@help added . . . . .	407		
\no@alphabet@error: Changed to error call . . . . .	407		

1990-04-02 ltfssini.dtx v1.1i		1991-08-14 ltpictur.dtx LaTeX2.09	
General: <code>\input</code> of files now handled by docstrip. . . . .	515	General: (RmS) inserted extra braces around entry for NFSS . . . . .	712
1990-04-05 lfsstrc.dtx v1.0m		1991-08-14 ltthm.dtx LaTeX2.09	
<code>\selectfont</code> : Call <code>\tracingon</code> only if <code>\tracingfonts</code> greater than 3. . . . .	459	<code>\@endtheorem</code> : Moved <code>\itshape</code> after <code>\item</code> to make it work with NFSS . . . . .	742
1990-05-05 lfsstrc.dtx v1.0n		1991-08-26 ltfssini.dtx v1.1n	
<code>\selectfont</code> : <code>\tracingon</code> with new syntax. . . . .	459	<code>\reset@font</code> : Macro introduced . . . . .	538
1990-06-23 ltfssini.dtx v1.1k		1991-08-26 ltmiscen.dtx LaTeX2.09	
<code>\nfss@text</code> : Changed to <code>\mbox</code> . . . . .	538	<code>\overline</code> : <code>\@par</code> added . . . . .	629
1990-06-24 ltfssbas.dtx v1.2j		1991-08-26 ltpictur.dtx LaTeX2.09	
<code>\DeclarePreloadSizes</code> : Missing percent added. . . . .	413	<code>\endpicture</code> : (RmS & FMi) extra boxing level around <code>\picbox</code> to guard against unboxing in math mode (proposed by John Hobby)	710
1990-06-24 lfsstrc.dtx v1.0o		1991-08-26 ltplain.dtx LaTeX209	
<code>\baselinestretch</code> : Moved to <code>tracefnt.dtx</code> . . . . .	464	<code>\tracingall</code> : Added <code>\errorcontextlines=\maxdimen</code> , suggested by J. Schrod . . . . .	35
<code>\getanddefine@fonts</code> : <code>\Adding</code> tracing code. . . . .	468	1991-09-29 ltboxes.dtx LaTeX2.09	
<code>\Macro</code> moved from <code>fam.dtx</code> . . . . .	467	<code>\@mpfootnotetext</code> : (RmS) added <code>\reset@font</code> . . . . .	678
Adding debug code. . . . .	467	1991-09-29 ltfloat.dtx LaTeX2.09	
<code>\use@mathgroup</code> : Tracing code added. . . . .	467	<code>\footnotetext</code> : (RmS) added <code>\reset@font</code> . . . . .	775
1990-06-30 ltfssbas.dtx v1.2l		1991-09-29 ltmath.dtx LaTeX2.09	
<code>\showhyphens</code> : Macro added. . . . .	430	<code>\@eqnnum</code> : RmS: <code>\reset@font</code> added. . . . .	644
1990-06-30 lfsstrc.dtx v1.0p		1991-09-29 ltsect.dtx LaTeX2.09	
<code>\use@mathgroup</code> : Added <code>\relax</code> after math group number. . . . .	467	<code>\@dottedtocline</code> : (RmS) added <code>\reset@font</code> for page number . . . . .	754
1990-07-07 lfsstrc.dtx v1.0q		1991-10-17 ltcntrl.dtx LaTeX209	
<code>\getanddefine@fonts</code> : Group number added to tracing. . . . .	468	<code>\@tfor</code> : (Rms) <code>\xdef</code> replaced by <code>\def</code> (See FMi's array.doc) . . . . .	278
<code>\math@egroup</code> : Tracing code added. . . . .	467	1991-10-25 ltbibl.dtx LaTeX2.09	
<code>\use@mathgroup</code> : Group number added to tracing. . . . .	467	<code>\@citex</code> : added <code>\reset@font</code> , suggested by Bernd Raichle. . . . .	784
1990-08-27 lfsstrc.dtx 1.0r		1991-11-01 ltfloat.dtx LaTeX2.09	
<code>\type@restoreinfo</code> : Some extra tracing info. . . . .	463	<code>\footnote</code> : (RmS) Added <code>\let\protect\noexpand</code> in <code>\footnote</code> , <code>\footnotemark</code> , and <code>\footnotetext</code> , since <code>\xdef</code> is used . . . . .	775
1990-08-27 lfsstrc.dtx v1.0r		1991-11-04 ltlists.dtx LaTeX2.09	
<code>\getanddefine@fonts</code> : Correcting missing name after <code>\tracingon</code> . . . . .	467	<code>\makelabel</code> : (RmS) added default definition for <code>\makelabel</code> , to produce an error message. . . . .	663
1991-03-28 ltfssini.dtx v1.1m		1991-11-04 ltplain.dtx RmS	
<code>\copyright</code> : Extra braces added. . . . .	538	General: Removed <code>\itemitem</code> since never needed/useful with L <sup>A</sup> T <sub>E</sub> X. . . . .	33
1991-03-30 ltfssini.dtx v1.2g		1991-11-06 ltbibl.dtx LaTeX2.09	
<code>\newfont</code> : Definition added. . . . .	537	<code>\@citex</code> : added code to remove a leading blank . . . . .	784
<code>\symbol</code> : Definition added. . . . .	537		
1991-07-24 ltmiscen.dtx LaTeX2.09			
<code>\verbatim</code> : Added <code>\penalty\interlinepenalty</code> to definition of <code>\par</code> so that <code>\samepage</code> works . . . . .	629		
1991-08-14 ltmath.dtx LaTeX2.09			
<code>\cases</code> : (RmS) inserted extra braces around entry for NFSS . . . . .	639		

1991-11-13	ltbibl.dtx	LaTeX2.09		avoid conflicts with other channels allocated by \newread . . . . .	80
	\@bibitem:	Changed counter enumi to enumiv, as it says in the comment above . . . . .	783		
1991-11-21	ltfssini.dtx	v1.10		\@xmpar: (RmS) added \global\@ignorefalse . . . . .	770
	\reset@font:	Added extra braces for robustness. . . . .	538	\end@float: (RmS) changed \@esphack to \@Espack . . . . .	764
		Changed to protected version of macro. . . . .	538		
1991-11-22	ltfloat.dtx	LaTeX2.09		\trivlist: RmS: added \nmbrrlistfalse . . . . .	659
	\footnote:	(RmS) Added \let\protect\noexpand in \xfootnote, \xfootnotemark, and \xfootnotetext . . . . .	775		
1991-11-22	ltlists.dtx	LaTeX2.09		\begin: Changed \@ignoretrue to \@ignorefalse (as documented)	621
	\@item:	(RmS) Changed second call to \makelabel to \unhbox\@tempboxa. Avoids problems with side effects in \makelabel and is more efficient.	663		
1991-11-27	ltfssbas.dtx	v1.3a		General: Renamed \text to \nfss@text to make it internal.	515
		General: All \family, \shape etc. renamed to \fontfamily etc. . . .	407		
1991-11-27	ltfssini.dtx	v1.2a		\extract@alph@from@version: Macro added. . . . .	429
		General: All \family, \shape etc. renamed to \fontfamily etc. . . .	515	\select@group: Added call to \extract@alph@from@version. . . . .	428
1992-01-06	ltfssini.dtx	v1.2c			
		General: added slitex code . . . . .	515	1992-07-26 ltfssbas.dtx v1.9a	
1992-01-10	ltbibl.dtx	LaTeX2.09		\curr@fontshape: . . . . .	420
	\@bibitem:	Changed \c@enumiv to \value of \@listctr . . . . .	783	\DeclareFontShape: Introduced \DeclareFontShape . . . . .	408
1992-01-10	ltmath.dtx	LaTeX2.09		\define@newfont: . . . . .	420
	equation:	RmS: put \hbox around \eqnnum to typeset the equation number in text mode (as in the eqnarray env.) . . . . .	644	\math@fonts: . . . . .	427
1992-01-10	ltthm.dtx	LaTeX2.09		\select@group: . . . . .	428
	\@othm:	(RmS) Check for existence of theorem environment . . . . .	741	\split@name: Added splitting into \f@encoding. . . . .	420
1992-01-14	ltbibl.dtx	LaTeX2.09		\wrong@fontshape: . . . . .	425
	\@biblabel:	removed \hfill . . . . .	786	1992-07-26 lfsstrc.dtx v2.0b	
1992-01-14	ltsect.dtx	0.0		\s@fct@: . . . . .	476
	\@starttoc:	(RmS) added \immediate to \openout as all \write commands are also executed \immediate . . . . .	752	\s@fct@sub: documentation fixes . . . . .	477
1992-02-26	ltbibl.dtx	LaTeX2.09		\selectfont: . . . . .	460
	\@lbibitem:	Added \hfill to restore left-alignment of bibliography labels in alpha style . . . . .	783	\try@simple@size: . . . . .	470
1992-03-18	ltdefns.dtx	LaTeX209		\try@size@range: . . . . .	474
	General:	(RMS) changed input channel from 0 to \inputcheck to		\use@mathgroup: . . . . .	467
				1992-08-14 ltbibl.dtx	LaTeX2.09
				\@citex: added missing argument braces around \hbox, found by Ed Sznyter . . . . .	784
				1992-08-14 ltboxes.dtx	LaTeX209
				\endminipage: (RmS) replaced \vskip-\lastskip by \unskip (proposed by FMI) . . . . .	677
				1992-08-17 ltbibl.dtx	LaTeX2.09
				\@citex: simplified code for removing leading blanks in citation key (proposed by Frank Jensen and Kresten Krab Thorup) . . . . .	784
				1992-08-19 ltsect.dtx	0.0
				\@xsect: (RmS) corrected bug: stretch and shrink in argument to \hskip	

previously not negated . . . . .	748	\footnote: (RmS) Changed all to ‘def’protect’noexpand’protect’noexpand . . . . .	775
1992-08-19 ltthm.dtx LaTeX2.09			
\@othm: (RmS) Changed error message to complain about undefined counter . . . . .	741	\hexnumber@: Make it accept counters. . . . .	538
1992-08-20 ltfssini.dtx v1.4b			
\@setsizE: Added \currsize. . . . .	537	1993-03-08 preload.dtx v2.0b General: Added 12pt preloads . . . . .	564
1992-08-24 ltdefns.dtx LaTeX209			
\@ifnextchar: (Rms) \@ifnextchar didn't work if its first argument was an equal sign. . . . .	102	1993-03-18 ltfssbas.dtx v2.0c General: Changed all \tempdima in \tempdimb to avoid killing \numberline . . . . .	407
1992-08-24 ltmiscen.dtx LaTeX2.09			
\begin: Added code to \begin to remember line number. Used by \badend to display position of non-matching \begin. . . . .	621	1993-03-18 lfsstrc.dtx v2.1b General: Changed all \tempdima in \tempdimb to avoid killing \numberline . . . . .	455
\verb: Changed \verb and \@sverb to work correctly in math mode . . .	634	Changed all \tempdimb in \tempdimx to avoid killing \numberline . . . . .	455
1992-08-25 ltsect.dtx LaTeX2.09			
\@sect: (FMi) replaced explicit setting of \@svsec by call to \@seccntformat . . . . .	747	1993-03-18 lfsstrc.dtx v2.1c \DeclareSizeFunction: Added all args to avoid blanks problems . . .	473
1992-09-18 ltlists.dtx LaTeX2.09			
\item: (RmS) Added warning if \item is used in math mode . . . . .	661	1993-04-09 lterror.dtx v1.0e \@latexerr: Mention The Companion . . . . .	285
1992-09-18 ltab.dtx LaTeX2.09			
\@array: Changed \par to \empty to avoid starting new row e.g. after \hline . . . . .	695	1993-04-11 lterror.dtx v1.0f \@latexerr: Remove setting of errorcontextlines . . . . .	285
1992-09-19 lfsstrc.dtx v2.0c			
\try@simple@size: . . . . .	470	1993-05-05 lftntcmd.dtx v2.0b General: Removed all LaTeX related cmds . . . . .	567
1992-09-21 ltfssini.dtx v1.4d			
\@not@math@alphabet: Macro defined.	536	1993-05-16 ltfssbas.dtx v2.0e \showhyphens: Use \reset@font . . .	430
1992-09-22 ltfssbas.dtx v1.91a			
General: Introduced \tf@size for math size. . . . .	407	1993-07-16 lfsstrc.dtx v2.1h General: Changed layout of info messages . . . . .	455
1992-09-22 lfsstrc.dtx v2.1a			
\getanddefine@fonts: Introduced \tf@size for math size. . . . .	468	1993-07-17 ltoutenc.dtx 1.0d General: changed \catcoding @ . . .	354
1992-11-13 ltfssini.dtx v?			
\hexnumber@: Made expandable. . . .	538	1993-08-03 ltmiscen.dtx LaTeX2.09 \enddocument: Changed redefinition of \global to redefinition of \@setckpt. . . . .	615
1992-11-23 ltcounds.dtx LaTeX209			
\stepcounter: Replaced {} in \stepcounter by \begingroup \endgroup to avoid adding an empty ord in math mode . . . . .	399	1993-08-05 ltpictur.dtx LaTeX2.09 \circle: (RMS) Added error message if \circle is used in math mode. . . . .	732
1992-11-26 ltboxes.dtx LaTeX2.09			
\@mpfootnotetext: (RmS) added protection for \edef . . . . .	678	1993-08-05 ltsect.dtx LaTeX2.09 \@sect: (RmS) Made sure that \protect works correctly in expansion of \the counter . . . . .	747
1992-11-26 lfloat.dtx LaTeX2.09			
\@footnotetext: (RmS) added protection for \edef . . . . .	775	1993-08-05 lspace.dtx LaTeX2e \@hspacE: (RmS) Removed superfluous \leavevmode in \hspace and \@hspacE, as suggested by CAR. . . . .	327

1993-08-05 lttab.dtx latex2e	\everycr . . . . .	712
	\tabular*: Replaced	
	\expandafter\def by \cnamedef.	694
1993-08-06 ltbibl.dtx LaTeX2.09	\citet: Moved writing to .aux file in	
	loop over citation keys so that	
	leading blanks are removed there	
	as well. . . . .	784
1993-08-13 ltoutenc.dtx 1.0f	General: Protected against active @	
	sign. . . . .	354
1993-08-13 preload.dtx v2.0c	General: Added \relax at end of font	
	names. . . . .	565
1993-08-16 ltoutenc.dtx 1.0g	General: Needs space after \string	354
1993-08-18 ltfssdcl.dtx v2.0e	\new@mathversion: Exchanged names	
	of encodings in warning message of	
	\SetSymbolFont. . . . .	496
1993-09-02 lfsstrc.dtx v2.1i	General: Corrected name of sgen size	
	function. . . . .	455
1993-09-03 ltmiscen.dtx LaTeX2.09	\verb@nolig@list: Replaced	
	\@noligs by extensible list . . . . .	634
1993-09-07 ltmiscen.dtx LaTeX2.09	\verb@balance@group: (RmS)	
	Changed definition of \verb so	
	that it detects a missing second	
	delimiter. . . . .	633
1993-09-08 ltmiscen.dtx LaTeX2.09	\enddocument: Added warning in case	
	of undefined references. . . . .	615
1993-09-15 ltfsbas.dtx v2.0g	\DeclareFontEncoding: Corrected:	
	\default@T to \default@M. . . . .	411
1993-09-15 lfsstrc.dtx v2.1j	General: Corrected spelling of	
	\noxpand. . . . .	455
1993-09-19 lterror.dtx LaTeX2.09	\@invalidchar: (RmS) Error message	
	for invalid input characters. . . . .	289
1993-11-02 ltmath.dtx LaTeX2.09	General: RmS: Corrected description	
	of \eqnse1, moved \eqnse1	
	accordingly and removed extra	
	\tabskip assignment. . . . .	644
1993-11-03 ltmath.dtx LaTeX2e	General: RmS: Initialized \everycr to	
	empty . . . . .	644
1993-11-03 ltpictur.dtx LaTeX2.09	General: (RmS) changed \halign to	
	\ialign to initialize \tabskip and	
	\everycr . . . . .	712
	\normalfont: Macro added . . . . .	538
1993-11-11 ltfssini.dtx v2.1a	\currext: Name changed from	
	\currextension . . . . .	796
1993-11-11 lfsstrc.dtx v2.2a	\reset@options: macro added . . . . .	822
	\AtEndDocument: Included extension	
	in the generated macro name for	
	package and class hooks. . . . .	822
\documentstyle: Added	\RequirePackage	
	\@unusedoptionlist stuff. . . . .	811
\load@onefilewithoptions: Moved	resetting of \default@ds, \ds@ and	
	\@declaredoptions here, from the	
	end of \ProcessOptions. . . . .	816
\NeedsTeXFormat: made more robust	for alternative syntax for other	
	formats. . . . .	813
\ProcessOptions*: Optimize ‘empty	option’ code. . . . .	808
	Stop adding the global option list	
	inside class files. . . . .	808
1993-11-14 ltdefns.dtx v0.2a	\g@addto@macro: Made global . . . . .	107
1993-11-15 ltclass.dtx v0.2b	\documentstyle: Modified to match	
	\ProcessOption*. . . . .	811
	\ProcessOptions*: Star form added. . . . .	808
1993-11-17 ltclass.dtx v0.2c	\@fileswith@ptions: Macro added	822
	\@badrequireerror: Macro added . . . . .	824
	\@twoloadclasserror: Macro added	824
\CurrentOption: Name changed from	\@curroption . . . . .	796
	\@curroption . . . . .	796
\DeclareOption*: Error checking	added . . . . .	806
\load@onefilewithoptions: Added	trap for two \LoadClass	
	commands. . . . .	818
\NeedsTeXFormat: Name changed from	\NeedsFormat . . . . .	813
	\ProcessOptions*: restoring	
	\fileswith@ptions added. . . . .	808
1993-11-18 ltclass.dtx v0.2d	\documentstyle: Modified	
	\RequirePackage stuff. . . . .	811
\ExecuteOptions: Use	\CurrentOption not \reserved@a	810

\NeedsTeXFormat: \fmtname		\newcommand: Macro reimplemented
\fmtversion not \C... . . . . .	813	and extended . . . . .
1993-11-21 ltfiles.dtx LaTeX2e		\renewcommand: Macro reimplemented
\@missingfileerror: Stop infinite		and extended . . . . .
looping on \Cer@ext . . . . .	347	\renewenvironment: Macro
1993-11-21 ltmiscen.dtx v0.9a		reimplemented and extended . . . . .
\@verbatim: use \verb@font		\two@digits: Macro added . . . . .
instead of \tt . . . . .	630	78
\verb: Use \verb@font instead of		1993-11-23 ltoutput.dtx v0.1a
\tt. . . . .	634	\paperheight: Register added . . . . .
\verb@font: Macro added . . . . .	630	910
1993-11-22 ltclass.dtx v0.2f		\paperwidth: Register added . . . . .
\@fileswithoptions: Made the		910
default [] not [\@unknownversion] . . . . .	814	1993-11-23 ltoutput.dtx v0.1c
\@ifl@ter: Added //00 so parsing		\enlargepage: Command added . . . . .
never produces a runaway		954
argument. . . . .	801	\kludgeins: Insert added . . . . .
General: \@unknownversion removed . . . . .	791	954
\load@onefilewithoptions: Made the		\makecol: Command changed . . . . .
initial version [] not		921
[\@unknownversion] . . . . .	816	\specialoutput: Command changed . . . . .
1993-11-22 ltdefns.dtx LaTeX2e		915
\@minus: Macro added . . . . .	79	\enlargethispage*: Commands
\@plus: Macro added . . . . .	79	added . . . . .
\CheckCommand: Macro added . . . . .	86	954
\providecommand: Macro added . . . . .	86	1993-11-24 ltfntcmd.dtx v2.1a
1993-11-22 lterror.dtx LaTeX2e		\maybe@ic@: Use \t@st@ic . . . . .
\c@errorcontextlines: Macro added . . . . .	285	572
1993-11-22 ltfiles.dtx LaTeX2e		\t@st@ic: Macro added . . . . .
\listfiles: Removed checking for		573
\@unknownversion . . . . .	349	1993-11-24 ltfssini.dtx v2.1a
1993-11-22 ltlenth.dtx LaTeX2e		General: Removed \xpt stuff . . . . .
\@settodim: Macro added . . . . .	405	538
\@settopoint: Macro added . . . . .	406	1993-11-24 ltlogos.dtx LaTeX2e
\settodepth: Macro added . . . . .	405	\LaTeX: Macro changed . . . . .
\settoheight: Macro added . . . . .	405	329
1993-11-22 ltlogos.dtx LaTeX2e		1993-11-28 ltclass.dtx v0.2h
\LaTeXe: Macro added . . . . .	329	\@twoclasseserror: Macro added . . . . .
1993-11-23 ltclass.dtx v0.2g		824
\@use@option: Name changed from		General: Assorted commands now in
\@executeoption . . . . .	810	the kernel removed. . . . .
General: Various macros now moved		795
to latex.tex. . . . .	795	Directory syntax checking moved to
Warnings and errors now directly		dircheck.dtx . . . . .
coded. . . . .	795	795
1993-11-23 ltdefns.dtx LaTeX2e		Primitive filenames now terminated
\@argdef: Macro added . . . . .	82	by space not \relax. . . . .
\@ifundefined: Redefined to remove a		795
trailing \fi . . . . .	101	\endfilecontents: Don't globally
\@newcommand: Macro added . . . . .	82	allocate a write stream (always use
\@newenv: Macro interface changed . . . . .	85	15) . . . . .
\@xargdef: Macro interface changed . . . . .	82	824
\@yargd@f: Avoid \C?@? token . . . . .	83	1993-11-28 ltfiles.dtx LaTeX2e
Macro interface changed . . . . .	83	\@missingfileerror: Use filename
		parser from dircheck . . . . .
		347
1993-11-30 ltoutput.dtx v1.0c		1993-11-29 ltoutput.dtx v1.0b
\f@tracemessage: Commands added . . . . .	956	\@makecol: \@makespecialcolbox
1993-12-01 fontdef.dtx v2.1a		added . . . . .
General: Update for LaTeX2e . . . . .	543	921
1993-12-01 ltoutput.dtx v1.0e		\@makespecialcolbox: Command
\@reinserts: Command added . . . . .	923	added . . . . .
1993-12-03 ltboxes.dtx v0.1a		1993-11-29 lplain.dtx LaTeX2e
\@argsbox: macro removed . . . . .	680	General: All accents in decimals;
		suggested by Paul Taylor . . . . .
		34

\@begin@tempboxa: macro added . . .	668	1993-12-05 ltfloat.dtx LaTeXe
\@end@tempboxa: macro added . . .	668	\@dblfloataplacement: Command changed . . . . .
\@iirsbox: redefined to support \height . . . . .	680	766
\@imakebox: macro modified . . . . .	668	\@xfloat: Command changed . . . . .
\@iirsbox: redefined to support \height . . . . .	680	760
\@isavebox: color support . . . . .	671	1993-12-05 ltoutput.dtx v1.0f
extra group . . . . .	671	\@addtobot: Command changed . . . . .
\@isavepicbox: extra group . . . . .	671	935
\@makebox: default changed from x to c . . . . .	668	\@addtocurcol: Command changed . . . . .
\@makepicbox: macro modified . . . . .	669	937
\@savebox: default c not x . . . . .	671	\@addtobblcol: Command changed . . . . .
\bm@b: macros added . . . . .	668	948
\endlrbox: macro added . . . . .	672	\@addtonextcol: Command changed . . . . .
\fbox: extra group . . . . .	672	944
\lrbbox: color support . . . . .	671	\@addtotoporbot: Command changed . . . . .
macro added . . . . .	671	936
\makebox: modified . . . . .	667	\@boxfpsbit: Command added . . . . .
\mbox: extra group . . . . .	668	960
\minipage: Redefined to support extra optional arguments . . . . .	676	\@flcheckspace: Command added . . . . .
\newsavebox: Pass the whole of arg 1 to \ifdefinable . . . . .	670	962
\parbox: Redefined to support extra optional arguments . . . . .	674	\@flsetnum: Command added . . . . .
\raisebox: redefined to support \height . . . . .	680	961
\sbox: color support . . . . .	671	\@flsettextmin: Command added . . . . .
extra group . . . . .	671	962
\set@color: color support . . . . .	670	\@flstop: Commands added . . . . .
macro added . . . . .	670	958
1993-12-03 ltclass.dtx v0.2i		\@flupdates: Command added . . . . .
\@cls@pkg: Name changed to avoid clash with output routine. . . . .	823	963
General: \@onlypreamble: Many commands declared. . . . .	795	\@fpsadddefault: Command added . . . . .
Removed obsolete \@documentclass . . . . .	795	959
1993-12-03 lterror.dtx v1.0b		\@getfpsbit: Command added . . . . .
\@latexerr: Set \c@errorcontextlines to -1 . . .	285	960
1993-12-03 ltfsini.dtx v2.1a		\@shipoutsetup: Command added . . . . .
General: update for LaTeXe . . . . .	515	925
1993-12-04 ltfilehook.dtx v0.9b		\@startcolumn: Command changed . . . . .
\unqu@tefilef@und: Macro added . . . . .	858	931
1993-12-04 ltfiles.dtx v0.9b		\@startdblcolumn: Command changed . . . . .
\@iinput: Macro reimplemented . . . . .	346	960
\@input: Macro reimplemented . . . . .	347	\@testfp: Command added . . . . .
\IfFileExists@: Macro added . . . . .	344	925
\input: Macro reimplemented . . . . .	346	\@textfloatsheight: Commands added . . . . .
1993-12-06 ltclass.dtx v0.2k		959
\ExecuteOptions: Preserve \CurrentOption . . . . .		915
1993-12-06 ltoutput.dtx v1.0f		1993-12-06 ltclass.dtx v0.2k
\@specialoutput: Unboxing of 255 added to rescue writes . . . . .		\@topnewpage: Commands changed . . . . .
1993-12-06 ltoutput.dtx v1.0g		913
\@topnewpage: \@floatplacement placement bug fixed . . . . .		\@tryfcolumn: Command changed . . . . .
1993-12-07 ltclass.dtx v0.2l		932
\ProvidesFile: Macro added . . . . .		\@writesetup: \@startpagehook added . . . . .
1993-12-07 ltclass.dtx v0.2m		925
\load@onefilewithoptions: Reset \CurrentOption . . . . .		\output: Command changed . . . . .
1993-12-07 ltoutenc.dtx 1.1		915
General: Protected all special characters with \string . . . . .		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: \@floatplacement placement bug fixed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: \@floatplacement placement bug fixed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913
		1993-12-07 ltclass.dtx v0.2m
		\@topnewpage: Commands changed . . . . .
		913
		\@tryfcolumn: Command changed . . . . .
		932
		\@writesetup: \@startpagehook added . . . . .
		925
		\output: Command changed . . . . .
		915
		1993-12-07 ltclass.dtx v0.2l
		\@floatplacement: Unboxing of 255 added to rescue writes . . . . .
		913

1993-12-07 ltoutenc.dtx v1.1		1993-12-11 ltmath.dtx v0.9g	
General: Made all character numbers decimal. ....	351	General: Added a group around the first argument of <code>\frac</code> to prevent changes (for example font changes) from modifying the contents of the second argument. ....	644
Removed a lot of equal signs and the like. ....	351		
1993-12-08 ltboxes.dtx v0.1b		1993-12-11 ltoutenc.dtx v1.2a	
<code>\@begin@tempboxa</code> : Extra braces for color support (braces removed from other macros) ....	668	General: Corrected for t1enc, math.	351
<code>\@irsbox</code> : fix typo ....	680	1993-12-11 ltsect.dtx LaTeXe	
<code>\@parboxto</code> : <code>\endgraf</code> added due to extra group in <code>\@begin@tempboxa</code> ....	675	<code>\author</code> : Added default ....	743
<code>\lrbbox</code> : move <code>\endpfalse</code> out of the inner group ....	671	<code>\title</code> : Added default ....	743
1993-12-08 lftntcmd.dtx v2.1b		1993-12-11 ltxref.dtx LaTeXe	
General: Macros <code>\rm</code> , <code>\bf</code> and <code>\sf</code> moved to classes.dtx ....	575	<code>\@setref</code> : Macro added ....	611
1993-12-08 lflists.dtx LaTeXe		<code>\@pageref</code> : Macro reimplemented ...	611
<code>\@item</code> : use <code>\sbox</code> to support colour ....	662	<code>\@ref</code> : Macro reimplemented ....	611
1993-12-08 ltspace.dtx LaTeXe		1993-12-12 ltoutput.dtx v1.0h	
<code>\@bsphack</code> : Command reimplemented Command reimplemented; late birthday present for Chris ....	317	<code>\@cflb</code> : <code>boxmaxdepth</code> setting moved defs changed to lets ....	929
<code>\@vbsphack</code> : Command added ....	320	<code>\@cflt</code> : name changed ....	929
1993-12-09 ltboxes.dtx v0.1c		<code>\@doclearpage</code> : defs changed to lets ....	920, 921
<code>\@irsbox</code> : fix another typo ....	680	<code>\@maketcol</code> : defs changed to lets ...	922
1993-12-09 ltclass.dtx v0.2n		<code>\@resethfps</code> : Warnings added: minimal ....	961
<code>\documentstyle</code> : input 209 compatibility file. ....	811	<code>\@startdblcolumn</code> : defs changed to lets ....	931, 932
1993-12-09 ltfiles.dtx v0.9e		<code>\@topnewpage</code> : braces removed ....	913
<code>\document</code> : Hook added ....	332	<code>\@tryfcolumn</code> : defs changed to lets ..	932
1993-12-09 ltmiscen.dtx v0.9e		<code>\fl@tracemessage</code> : Commands changed ....	956
<code>\enddocument</code> : Hook added ....	615	1993-12-13 ltclass.dtx v0.2o	
1993-12-10 ltoutenc.dtx v1.2		General: Removed setting <code>\errorcontextlines</code> (now in latex.tex) ....	795
General: Added source code for t1enc.sty. ....	351	<code>\documentstyle</code> : compatibility file now latex209.sty. ....	811
1993-12-11 lftntcmd.dtx v3.0a		<code>\usepackage</code> : Fixed error handling ..	813
General: Complete reworking of all text commands, using just one creator function ....	567	1993-12-13 ltdirchk.dtx v0.2a	
italic correction now put in front of penalty before glue ....	567	General: on the 'docstrip' pass, do not check openin path ....	10
newcommands replaced by defs ..	567	<code>\IfFileExists</code> : Removed interactive prompting for current directory syntax ....	10
newfontswitch command corrected and changed ....	567	<code>\strip@prefix</code> : modified, name changed from <code>\stripmeaning</code> . ...	5
<code>\DeclareTextFontCommand</code> : Macro changed ....	569	1993-12-13 lflists.dtx latex2e	
<code>\emph</code> : Macro changed ....	570	<code>\trivlist</code> : Initialised <code>\@itemlabel</code>	659
<code>\fix@penalty</code> : Macro added ....	573	1993-12-13 ltmiscen.dtx v0.9h	
<code>\maybe@ic</code> : Macro name changed ..	572	<code>\@noligs</code> : Readded <code>\@noligs</code> ....	634
<code>\maybe@ic@</code> : Macro and name changed ....	572	<code>\@verbatim</code> : Readded <code>\@noligs</code> ... Removed optional argument of <code>\item</code> ....	630, 629
<code>\sw@slant</code> : Macro changed ....	573	<code>\center</code> : Removed optional argument of <code>\item</code> ....	627
<code>\textup</code> : Macros changed ....	570		

<b>flushleft:</b> Removed optional argument of \item .....	628	Removed the catcode hackery, since the file is only read as a package in the preamble, and removed all the messages on the screen, which just confuse users. Replaced them by the appropriate \ProvidesPackage commands. Added XXXenc. . . . .	354
<b>flushright:</b> Removed optional argument of \item .....	628		
1993-12-13 ltoutenc.dtx v1.2b			
General: Corrected file name in driver code. . . . .	351		
1993-12-13 lttab.dtx latex2e			
\tabbing: Removed optional argument of \item .....	689		
1993-12-14 ltoutput.dtx v1.0i			
General: Section added to declare all parameters . . . . .	968		
1993-12-15 ltboxes.dtx v0.1d			
\@imminipage: Changed default from ‘c’ to ‘s’ . . . . .	677		
\@iparbox: Changed default from ‘c’ to ‘s’ . . . . .	674		
\minipage: Changed default from ‘c’ to ‘s’ . . . . .	676		
extra space removed. . . . .	676		
\parbox: Changed default from ‘c’ to ‘s’ . . . . .	674		
1993-12-15 ltclass.dtx v0.2p			
General: Removed extra ‘s’ from \@warnings . . . . .	795		
1993-12-16 ltlogos.dtx LaTeX2e			
\LaTeXe: Extended logo by DPC . . . . .	329		
1993-12-16 ltmath.dtx v0.9i			
\@eqnrcr: use \refstepcounter instead of shortcut . . . . .	646		
General: use \refstepcounter instead of shortcut . . . . .	644		
1993-12-16 ltmiscen.dtx v0.9i			
General: \literal added . . . . .	634		
1993-12-16 ltpage.dtx LaTeX2e			
\mark: Init \mark at begin document . . . . .	789		
1993-12-16 ltspace.dtx LaTeX2e			
\@bsphack: Corrected optimisation :-) . . . . .	317		
1993-12-16 lttab.dtx latex2e			
\@xhline: Measure from middle of vertical rules . . . . .	704		
1993-12-17 ltclass.dtx v0.2q			
\@documentclasshook: Macro added . . . . .	795		
\@fileswithoptions: Add \@compatibility hook . . . . .	814		
\documentstyle: Match Alan’s new code. . . . .	811		
1993-12-17 ltoutenc.dtx 1.3			
General: Added this section . . . . .	355		
Removed all the hackery for use in \DeclareFontEncoding, and redid everything using \DeclareTextFoo. . . . .	367, 369		
1993-12-17 ltoutenc.dtx v1.3			
General: Added \EncodingSpecificAccent, \EncodingSpecificAccentedLetter and \EncodingSpecificCommand. . . . .	351		
1993-12-17 ltoutput.dtx v1.0j			
\@copcol: Hook removed . . . . .	921		
\@specialoutput: Page room test added . . . . .	916		
\@topnewpage: check for vsizetoo small added . . . . .	913		
Page room test added . . . . .	915		
\@writesetup: —and then removed . . . . .	925		
\f@l@tracemessage: tracefloatvals made a document command . . . . .	956		
1993-12-17 ltpage.dtx LaTeX2e			
\mark: Removed init \mark at begin document, since it doesn’t work. . . . .	789		
\rightmark: Stopgap solution to mark \leftmark and \rightmark work without initializing mark until the problem is solved. . . . .	789		
1993-12-18 ltoutenc.dtx 1.3b			
General: Fixed typos with \ProvidesPackage lines. Added the \NeedsTeXFormat line. Added the last argument to \DeclareEncoding. Moved the use of the encodings to after their declaration. . . . .	354		
Replaced the missing last argument to \DeclareFontEncoding. . . . .	367, 369		
1993-12-18 ltoutenc.dtx 1.3c			
General: Rewrote for the new syntax of \EncodingSpecific. . . . .	367, 369		
Split \EncodingSpecificAccent up into \EncodingSpecific and \DeclareAccent. . . . .	355		
1993-12-18 ltoutenc.dtx v1.3a			
General: Replaced OT3 by XXX . . . . .	351		
1993-12-18 ltoutenc.dtx v1.3b			
General: Corrected typos. . . . .	351		
Replaced the missing last argument to \DeclareFontEncoding. . . . .	351		

1993-12-18 ltoutenc.dtx v1.3c		1994-01-17 ltclass.dtx v0.2s	
General: A new syntax, separating accent-definitions from encoding-specific definitions, and allowing encoding-specific \chardef, \let, etc. . . . .	351	\@fileswithoptions: Modify to reduce parameter stack usage . . .	814
Rewrote for the new syntax of \EncodingSpecific. . . . .	351	General: Added many more \@onlypreamble commands . . . .	795
1993-12-18 ltoutenc.dtx v1.3d		Wrapped long lines to column 72	795
General: Some T1 stuff had drifted into the OT1 file. . . . .	351	\load@onefile@withoptions: Modify to reduce parameter stack usage	819
1993-12-18 ltpage.dtx LaTeXe		1994-01-17 ltfiles.dtx LaTeXe	
\sloppy: Added \emergencystretch	789	\listfiles: New Version, adds ‘tex’ if needed, and lines up columns .	349
1993-12-19 ltclass.dtx v0.2r		1994-01-17 ltfssbas.dtx v2.1a	
\endfilecontents: Different message when ignoring a file . . . . .	824	General: New math font setup . . .	407
1993-12-19 lftntcmd.dtx v3.0b		\curr@math@size: New math font setup . . . . .	419
General: \@pdef command added . .	567	\everydisplay: New math font setup	419
Added by ASAJ. . . . .	575	\everymath: New math font setup .	419
Made \@newfontswitch produce an error if command already exists, and added \@renewfontswitch, ASAJ . . . . .	567	\frozen@everydisplay: New math font setup . . . . .	419
Other tidying . . . . .	567	\frozen@everymath: New math font setup . . . . .	419
Some more tidying done . . . . .	567	\math@version: New math font setup	418
Untidying added, so this is now a TEMPORARY version. . . . .	567	1994-01-17 ltfssini.dtx v2.1e	
Wording changes by CAR. . . . .	575	\not@math@alphabet: Message changed . . . . .	536
\DeclareOldFontCommand: Corrected and tidied . . . . .	574	1994-01-17 lfsstrc.dtx v2.3a	
\DeclareTextFontCommand: Corrected and tidied . . . . .	569	General: New math font setup . . .	455
1993-12-19 ltspace.dtx LaTeXe		\check@mathfonts: New math font setup . . . . .	465
\@bsphack: There seem to be problems with selfmade birthday presents .	318	\glb@currscale: New math font setup	463
1993-12-20 ltdefs.dtx LaTeXe		\restglb@settings: New math font setup . . . . .	466
\@reargdef: Kept old version of \@reargdef, for array.sty . . . .	83	1994-01-18 ltbibl.dtx LaTeXe	
1993-12-20 ltfiles.dtx v0.9m		\bibliography: Use \@input@ so include files are listed. . . . .	784
\@obsoletefile: Added this command, removed @oldfilewarning . . . . .	349	1994-01-18 ltclass.dtx v0.2t	
1994-01-05 fontdef.dtx v2.1d		\@ifclassloaded: Fix typo \@pkgetension . . . . .	800
General: Removed nf prefix from file names. . . . .	545	1994-01-18 ltfilehook.dtx v0.9p	
1994-01-13 ltmath.dtx v0.9o		\unqu@tefilef@nd: New Definition	858
\@eqnacr: correcting 0.9i . . . .	646	1994-01-18 ltfiles.dtx v0.9p	
General: correcting 0.9i . . . .	644	\@iffileonpath: Macro added . . .	345
1994-01-14 ltdirchk.dtx v0.2d		\@input: do not use a different definition for \input@path . . . .	347
\IfFileExists: Close the texsys.aux output stream . . . . .	10	\@input@: Macro added . . . . .	347
1994-01-15 ltfiles.dtx v0.9o		\IfExists@: New Definition . .	344
\document: move \@preamblecmds after document hook . . . . .	334	\includeonly: Use \@input@ so include files are listed. . . . .	338

Only add <code>\penalty</code> if in hmode . . . . .	629	<code>\restglb@settings</code> : Correct trace info placement . . . . .	466
1994-01-19 fontdef.dtx v2.1e			
General: Added missing setting for symbols in bold version. . . . .	550	<code>\nocorrlist</code> : Only ., used as default for cm fonts . . . . .	574
1994-01-19 ltdirchk.dtx v0.2e			
<code>\IfFileExists</code> : name changed from <code>\test</code> . . . . .	9	1994-01-29 ltclass.dtx v0.2v	
<code>\input@path</code> : No longer check that an empty group is in the path . . . . .	11	<code>\@unprocessedoptions</code> : Macro added. . . . .	823
<code>\strip@prefix</code> : name changed from <code>\strip@meaning</code> , to match NFSS. . . . .	5	<code>\load@onefile@withoptions</code> : All options raise error if no <code>\ProcessOptions</code> appears . . . . .	820
1994-01-19 ltmath.dtx v1.0n classes			
<code>\mathindent</code> : Deferred setting of <code>\mathindent</code> . . . . .	647	1994-01-31 ltdefns.dtx v0.2w	
1994-01-20 ltdirchk.dtx v0.2f		<code>\g@addto@macro</code> : Use toks register to avoid ‘hash’ problems . . . . .	107
General: <code>\@copytexsys</code> and the <code>texsys.new</code> file removed . . . . .	9	1994-01-31 ltfiles.dtx v0.9t	
Modify all of ltxcheck . . . . .	14	<code>\document</code> : set <code>\@normalsize</code> or <code>\normalsize</code> if necessary . . . . .	333
<code>\IfFileExists</code> : <code>\@copytexsys</code> removed . . . . .	10	1994-01-31 ltdirchk.dtx v3.1b	
1994-01-21 ltclass.dtx v0.2u		General: <code>\@normalsize</code> no longer defined . . . . .	567
<code>\documentstyle</code> : compatibility file now latex209.def. . . . .	811	1994-02-01 ltpage.dtx LaTeX2e	
1994-01-21 ltdirchk.dtx v0.2g		<code>\pagestyle</code> : (DPC) Modify to get nicer error message . . . . .	787
General: Improve documentation, reorganize docstrip module . . . . .	1	<code>\thispagestyle</code> : (DPC) Modify to get nicer error message . . . . .	788
<code>\filename@parse</code> : Minor changes, and add Mac version () . . . . .	11	1994-02-02 ltclass.dtx v0.2x	
<code>\today</code> : Name changed from <code>\stamp</code> , to save memory . . . . .	9	<code>\load@onefile@withoptions</code> : Only run the hook and options check if the file was loaded. . . . .	820
1994-01-21 ltfloat.dtx LaTeX2e		1994-02-03 ltoutput.dtx v1.0k	
<code>\@xfloat</code> : Added missing percent characters. . . . .	760	<code>\makespecialcolbox</code> : correct mistakes in the documentation . . . . .	924
1994-01-21 ltmiscen.dtx v0.9s		1994-02-07 ltclass.dtx v0.2y	
<code>\verbatim@font</code> : Removed unnecessary category code hackery. . . . .	630	<code>\files@withoptions</code> : Run <code>\compatibility</code> on the first class to start (not the first to finish) . . . . .	814
1994-01-24 ltdirchk.dtx v0.2h		<code>\@ifclasswith</code> : Add extra ,s so ‘two’ is not matched with ‘twocolumn’ . . . . .	802
<code>\IfFileExists</code> : Stop testing once <code>texsys.aux</code> has been found . . . . .	10	<code>\ProcessOptions*</code> : Add extra ,s so ‘two’ is not matched with ‘twocolumn’ . . . . .	808
1994-01-24 ltpage.dtx LaTeX2e		1994-02-07 ltfssbas.dtx v2.1c	
<code>\pagestyle</code> : (DPC) Complain if <code>pagestyle</code> is undefined. . . . .	787	<code>\DeclareFontEncoding</code> : revert catcode settings earlier . . . . .	411
1994-01-25 ltdirchk.dtx v0.2i		<code>\DeclareFontShape@</code> : revert catcode settings earlier . . . . .	408
General: Protect against looping on <code>\@input</code> and <code>\@end</code> . . . . .	3	1994-02-08 ltoutput.dtx v1.0k	
1994-01-25 ltfssbas.dtx v2.1b		<code>\makespecialcolbox</code> : <code>boxmaxdepth</code> setting added . . . . .	924
<code>\math@version</code> : Corrections for math setup . . . . .	418	<code>boxmaxdepth</code> setting removed . . . . .	923
1994-01-25 ltmath.dtx LaTeX2e		General: Documentation and tasks tidied. . . . .	898
<code>\bordermatrix</code> : Removed <code>\p@renwd</code> . . . . .	639		
1994-01-26 lfsstrc.dtx v2.3c			
<code>\check@mathfonts</code> : Correct trace info placement . . . . .	465		

1994-02-10 ltclass.dtx v0.2z		1994-03-07 ltboxes.dtx v0.1a	
\@documentclasshook: Changed the name from \compatibility to \@documentclasshook, and added the check for whether \@normalsize has been defined. ASAJ. . . . .	795	\@mpfootnotetext: Extra group for color . . . . .	678
\@fileswithoptions: Renamed \compatibility to \@documentclasshook. ASAJ. . .	814	1994-03-07 ltboxes.dtx v1.0a	
1994-02-10 ltfssbas.dtx v2.1d		General: Unify format with other Kernel files . . . . .	667
\addto@hook: Made \addto@hook long. . . . .	432	1994-03-07 ltdefns.dtx v1.0a	
1994-02-10 ltfsscmp.dtx v2.1d		\@ccitaliccorr: Macro added . . . . .	79
\scan@fontshape: scan away stuff after pt . . . . .	481	1994-03-07 ltfiles.dtx v1.0a	
1994-02-22 ltfssini.dtx v2.1g		General: Initial version, split from latex.dtx . . . . .	330
General: Correct error message . . .	540	Long lines wrapped to 72 columns	330
1994-02-24 ltfssbas.dtx v2.1e		1994-03-07 ltfinal.dtx v0.1a	
\DeclareFontShape: Separate restoration of catcodes for fd cmds . . . . .	408	General: Add code from the old dump.dtx . . . . .	988
\define@newfont: Separate restoration of catcodes for fd cmds . . . . .	421	Initial version, split from latex.dtx	974
\@fss@catcodes: Separate restoration of catcodes for fd cmds . . . . .	421	move code here from lhyphen.dtx	980
1994-02-25 ltdirchk.dtx v0.2j		Remove oldcomments environment	974
General: Remove need for drv file . . .	1	use \InputIfFileExists not \IfFileExists . . . . .	980
1994-03-01 ltdirchk.dtx v0.2k		1994-03-07 ltfloat.dtx v1.0a	
General: Add unstripped module, so that dircheck.dtx may be used with initex . . . . .	1	\@endfloatbox: (DPC) Extra group for colour . . . . .	765
1994-03-02 ltboxes.dtx v0.1e		\@footnotetext: (DPC) Extra group for colour . . . . .	775
General: Add 2ekernel module . . . . .	667	\@xfloat: (DPC) Extra group for colour . . . . .	761
Remove need for drv file . . . . .	667	1994-03-07 lthyphen.dtx v0.1c	
1994-03-02 ltclass.dtx v0.3a		General: move the 2ekernel code to ltfinal.dtx . . . . .	972
General: Remove need for driver file . . .	795	1994-03-07 ltlength.dtx v1.0a	
1994-03-03 ltboxes.dtx v0.1f		\@settdim: (DPC) Extra group for colour . . . . .	405
\@irsbox: Replaced a missing \else . . .	680	1994-03-07 ltlists.dtx v1.0a	
1994-03-04 ltfloat.dtx v1.0a		General: Initial version, split from latex.dtx . . . . .	651
General: Initial version, split from latex.dtx . . . . .	756	Long lines wrapped to 72 columns	651
1994-03-04 ltsect.dtx v1.0a		1994-03-07 ltpage.dtx v1.0a	
General: Initial version, split from latex.dtx . . . . .	743	General: Initial version, split from ltherest.dtx . . . . .	787
1994-03-04 lttab.dtx v1.0a		1994-03-07 ltpictur.dtx v0.1a	
General: Initial version, split from latex.dtx . . . . .	682	General: Initial version, split from latex.dtx . . . . .	707
1994-03-04 ltvers.dtx v1.0a		Long lines wrapped to 72 columns	707
General: Initial version, split from latex.dtx . . . . .	39	1994-03-07 ltsect.dtx v1.0a	
		\@hangfrom: (DPC) Extra groups for colour . . . . .	750
		1994-03-07 lttab.dtx v1.0a	
		General: Long lines wrapped to 72 columns . . . . .	682
		1994-03-08 ltclass.dtx v0.3b	
		General: Modify driver code into ‘new style’ . . . . .	795

1994-03-08 ldirchk.dtx v1.0a	\listfiles: Reset \@addtofilelist at begin document .....	350	
General: Reorganize driver module into 'new style' .....	1	1994-03-13 ltfssbas.dtx v2.1g	
1994-03-08 lplain.dtx v1.0a	General: add 2ekernel module to omit repeated code .....	407	
General: Remove need for a driver file.	15	1994-03-13 ltfssdcl.dtx v2.1c	
1994-03-10 ltfssbas.dtx v2.2f	General: add 2ekernel module to omit repeated code .....	485	
\mathat@egroup: Changed	430	1994-03-14 ltboxes.dtx v1.0b	
\begin{group}/\end{group} to	\@isavebox: Use \color@setgroup .....	671	
\bgroup/\egroup. ....	\@isavepicbox: Use	671	
1994-03-11 ltfssdcl.dtx v2.1b	\color@setgroup .....	671	
\DeclareSymbolFontAlphabet@:	\color@begingroup: macro added for color support .....	670	
Added check against use of alphabet switch outside of math mode. ....	\color@endgroup: macro added for color support .....	670	
\SetMathAlphabet@: Changed parameter template in temporary macro to catch check add below.	513	\lrbox: Use \color@setgroup .....	671
502	\sbox: Use \color@setgroup .....	671	
1994-03-12 ltclass.dtx v0.3c	1994-03-14 ltfloat.dtx 1.0c		
General: Change name from docclass to ltclass .....	795	\@xympar: (DPC) Use	
\ProvidesFile: Add \wlog .....	805	\color@begingroup .....	770
\ProvidesPackage: Add \wlog .....	803	1994-03-14 ltfloat.dtx v1.0c	
use \@tempa .....	803	\@endfloatbox: (DPC) Use	
1994-03-12 ltdefns.dtx v1.0b	\color@endgroup .....	765	
\@reargdef: New defn, in terms of	\@footnotetext: (DPC) Use		
\@argdef .....	\color@begingroup, add		
\@yargd@f: Name changed from	\endgraf .....	775	
\XXX@argdef .....	\@savemarbox: (DPC) Use		
1994-03-12 ldirchk.dtx v1.0b	\color@begingroup .....	769	
General: Change name from	\@xfloat: (DPC) Use		
dircheck.dtx .....	\color@begingroup .....	761	
Minor edits to the typeouts in	1994-03-15 ltfiles.dtx LaTeX2e		
ltxcheck .....	\@missingfileerror: Quit on x or X		
1994-03-12 ltfloat.dtx v1.0b	just like a real error .....	347	
\@savemarbox: (DPC) Extra group for colour .....	1994-03-15 ltftcmd.dtx v3.2a		
769	General: Adapted to mass formatting .....	567	
\@xympar: (DPC) Extra bgroup for colour .....	Changed \V/ to \@italiccorr .....	567	
770	Removed \@renewfontswitch .....	567	
1994-03-12 lplain.dtx v1.0b	Removed defs of short-forms and all sizes except \normalize .....	567	
General: Name changed from lplain.	1994-03-15 ltoutput.dtx v1.0l		
The end of an era .....	\@addtocurcol: Changed \addvspace to \vskip .....	943	
1994-03-12 lplain.dtx v1.0e	\@combinedblfloats: Removed boxmaxdepth setting. ....	930	
General: Replaced remaining width, height, depth by L <sup>A</sup> T <sub>E</sub> X macro names to save tokens. ....	\@makecol: \maxdepth changed to \@maxdepth .....	922	
15	Removed boxmaxdepth setting. ....	922	
1994-03-13 ltentrl.dtx v1.0c	\@makespecialcolbox: Removed boxmaxdepth setting. ....	924	
\@tfor: (DPC) Add \tfor so a single group is correctly treated. ....	\@topnewpage: Corrected and amended warning message .....	914	
278	Warning added: it should be improved .....	915	
1994-03-13 ltfilehook.dtx v0.3b			
\unqu@tefilef@nd: Use new cmd			
\@addtofilelist .....			
858			
1994-03-13 ltfiles.dtx LaTeX2e			
\@addtofilelist: Macro added .....			
349			

General: Added some warnings when page gets full of top floats. ....	898	1994-03-29 ltcnts.dtx v1.0c General: Create file from parts of ltmiscen and ltherest. ....	397
Driver added and further tidying.	898	1994-03-29 ltlength.dtx v1.0c General: Create file ltcntlen from parts of ltmiscen and ltherest. ....	405
Removed duplicated code and corrected docstrip options. ....	898	1994-03-29 ltmiscen.dtx v1.0d General: Remove counter macros to ltcntlen .....	614
Some boxmaxdepth settings removed. ....	898	1994-03-29 ltpageno.dtx v1.0c General: Create file ltcntlen from parts of ltmiscen and ltherest. ....	608
1994-03-16 ltclass.dtx v0.3f General: Add pkgindoc package ...	840	1994-03-29 ltxref.dtx v1.0c General: Create file ltcntlen from parts of ltmiscen and ltherest. ....	609
1994-03-16 ltfiles.dtx LaTeX2e \listfiles: Move this code directly into \document .....	350	1994-03-31 ltbibl.dtx v1.0a General: Initial version of ltidxbib.dtx, split from ltherest.dtx .....	782
1994-03-16 ltfiles.dtx v1.0c \document: (DPC) directly add file list settings .....	334	1994-03-31 ltidxglo.dtx v1.0a General: Initial version of ltidxbib.dtx, split from ltherest.dtx .....	779
1994-03-16 ltmiscen.dtx v1.0b \verbatim: Remove \global\@inlabelfalse again. .	629	1994-04-09 ltcnts.dtx v1.0d \newctr: \@nocnterr now has counter name argument .....	398
1994-03-28 ltalloc.dtx v1.0d General: Redefinition of 'new' allocations removed. ....	273	\addtocounter: \@nocnterr now has counter name argument .....	398
1994-03-28 ltdirchk.dtx v1.0d General: Improve documentation ...	1	\setcounter: \@nocnterr now has counter name argument .....	398
1994-03-28 lterror.dtx v1.0d \@invalidchar: (DPC) Comment out (use catcode15 instead) .....	289	\stepcounter: Use \addtocounter to have name checked .....	399
General: Remove test for \inputlineno undefined. ....	285	1994-04-09 lthm.dtx v1.0b \othm: Use standard counter error message (FMi) .....	741
1994-03-28 ltfiles.dtx v1.0d \document: (DPC) Use \normalsize not \@normalsize .....	333	1994-04-11 ltclass.dtx v0.3g \endfilecontents: Add star form, don't write \endinput at the end of the file. ....	824
(DPC) remove \@normalsize check .....	333	\ProvidesFile: Protect against weird catcodes. ....	805
1994-03-28 ltfloat.dtx v1.0b \caption: Use \normalsize not \@normalsize .....	759	1994-04-11 ltfssbas.dtx v2.1h General: Added \defaultscriptratio and \defaultscriptscriptratio. ASAJ. ....	407
General: Split further from ltherest.dtx .....	756	\defaultscriptratio: Macro added .....	430
1994-03-28 ltlists.dtx v1.0b General: Improve documentation ...	650	\defaultscriptscriptratio: Macro added .....	430
1994-03-28 ltmiscen.dtx v1.0c General: Improve Documentation ...	614	1994-04-12 ltboxes.dtx v1.0c General: Remove \@acci, now defined in ltplain.dtx .....	675
1994-03-28 ltplain.dtx v1.0c \newlanguage: Remove some \outer declarations. ....	18	Remove \@dischyp, now defined in ltinit.dtx .....	675
1994-03-28 ltsect.dtx v1.0b General: Split further from ltherest.dtx .....	743	1994-04-12 ltdefns.dtx v1.0g \dischyp: Define \@dischyp, was previously in ltboxes.dtx .....	105
1994-03-28 ltab.dtx v1.0b General: Improve documentation ...	682		
1994-03-28 ltthm.dtx v1.0a General: Initial version, split from latex.dtx .....	739		

1994-04-12 ltplain.dtx v1.0d		\fontsubfuzz: Changed dimen to macro . . . . .	474
General: Define \@acci . . . . .	34		
1994-04-12 ltvers.dtx v1.0b		\subst@size: \font@submax and \fontsubfuzz now macros . . . . .	475
General: Have version info generated automatically . . . . .	39		
1994-04-14 lftntcmd.dtx v3.2b			
General: Macros renamed to non-private forms, JB . . . . .	567		
\DeclareOldFontCommand: Renamed from \@newfontswitch . . . . .	574		
1994-04-15 ltboxes.dtx v1.0d			
\@isavebox: Added missing percent character . . . . .	671		
1994-04-17 ltcnts.dtx v1.0e			
\@newctr: Use \@nocounterr instead of \@nocnterr . . . . .	398		
\addtocounter: Use \@nocounterr instead of \@nocnterr . . . . .	398		
\setcounter: Use \@nocounterr instead of \@nocnterr . . . . .	398		
1994-04-17 lterror.dtx v1.0h			
\@nocnterr: New name for error message, old error message (without arg) kept . . . . .	286		
1994-04-17 ltthm.dtx v1.0c			
\@othm: Use new std counter error message (FMi) . . . . .	741		
1994-04-18 ltfinal.dtx v0.1b			
General: Initialise \textheight, \textwidth and page style . . . . .	977		
1994-04-18 ltfloat.dtx v1.0d			
\@footnotetext: (DPC) Remove Colour support . . . . .	775		
\@savemarbox: (DPC) Remove Colour support . . . . .	769		
1994-04-18 ltfssbas.dtx v2.1i			
General: Macro \no@alphabet@help removed again . . . . .	407		
\calculate@math@sizes: Changed message to log only . . . . .	430		
\no@alphabet@error: Use std LaTeX error macro . . . . .	407		
1994-04-18 ltfssdcl.dtx ???			
\DeclareMathAlphabet: Pass correct arg (2 not 3) . . . . .	500		
1994-04-18 ltfssdcl.dtx v2.1d			
General: Removed surplus \no@alphabet@error (see fam.dtx) . . . . .	485		
1994-04-18 lfsstrc.dtx v2.3d			
General: Changed to new error/warning scheme . . . . .	455		
\font@submax: Changed dimen to macro . . . . .	474		
		\enddocument@kernel@warnings: Changed logic for producing warning messages . . . . .	617
1994-04-21 ltboxes.dtx v1.0e			
\@iiminipage: Extra \bgroup for color . . . . .	677		
\@mpfootnotetext: Extra \endgraf for color . . . . .	678		
\endminipage: Extra \egroup for color . . . . .	677		
1994-04-21 ltfinal.dtx v0.1c			
General: Added comments, set the catcodes of 128–255 . . . . .	974		
1994-04-22 ltfssini.dtx v2.1g			
\not@math@alphabet: Message changed again . . . . .	536		
1994-04-23 ltfinal.dtx v0.1d			
General: Check that \font@submax is still zero . . . . .	974		
1994-04-24 ltoutput.dtx v1.0m			
\@resetfps: Number 2 changed to \tw@ . . . . .	961		
Warning changed . . . . .	961		
\@specialoutput: Message changed to give more info and ‘top’ removed . . . . .	916		
\@topnewpage: Message changed to give more info . . . . .	915		
Warning message removed as it will be generated later . . . . .	914		
General: Changed \normalsize to \normalsize. . . . .	898		
Corrected unverbed commands in documentation . . . . .	898		
Removed some long lines and other aesthetic changes . . . . .	898		
Warning messages changed/corrected . . . . .	898		
1994-04-24 ltpictur.dtx v0.1b			
General: Removed surplus spaces after \hbox to in several cases . . . . .	707		

1994-04-25 ltclass.dtx v0.3h	Warning changed to info message in \@protecteddef ..... 567
General: Removed spurious extra `s at the end of error messages .....	795
1994-04-25 ltfloat.dtx v1.0e	\@largefloatcheck: Changed warning message to give more info ..... 765
	Command added ..... 765
	General: Changed warning messages 756
	Removed obsolete tracing code .. 756
1994-04-27 lfsstrc.dtx v2.3e	General: Corrected item that was forgotten in last change. .... 455
1994-04-28 lterror.dtx v1.0j	\@inmatherr: Macro added ..... 288
1994-04-28 lterror.dtx v1.1c	\@inmatherr: Replaced \noexpand with \protect. .... 288
1994-04-28 ltssdcl.dtx v2.1e	General: Removed all \uppercase in hex num parsing macros ..... 485
1994-04-28 ltlists.dtx v1.0c	\item: Replaced \@ltxnomath by \@inmatherr ..... 661
1994-04-28 ltpictur.dtx v0.1c	\@multiput: (DPC) Macro added .. 711
	General: bezier curves added ..... 734
	\multiput: (DPC) Ignore spaces between )( ..... 710
	\picture: (DPC) Ignore spaces before ( ..... 709
1994-04-28 lplain.dtx v1.0g	General: Turn off overfull box tracing in log ..... 27
1994-04-29 ltclass.dtx v1.0a	General: Change version number to 1 (no other change) ..... 795
1994-04-29 ltmiscen.dtx v1.0f	\Overbatim: \leavevmode added ... 630
	Change to \everypar added ..... 630
1994-04-29 ltoutenc.dtx 1.4a	General: Removed \EncodingSpecific. Renamed all the commands. Added \DeclareTextGlyph and \UndeclareTextCommand. .... 355
	Removed Rokicki's OT1 variant encoding. Moved the driver to the top. .... 354
1994-04-30 lfntcmd.dtx v3.3b	General: Documentation up-dated and tidied ..... 567
	Prefix frag@ changed to frag in \@protecteddef ..... 567
	Title changed ..... 567
1994-04-30 ltoutput.dtx v1.0n	Warning changed to info message in \@protecteddef ..... 567
	\@activechar@info: \@activechar@warning changed to \@activechar@info ..... 925
	\@combinedblfloats: Removed rule in topnewpage case ..... 930
	\@emptycol: Empty column action added: \@emptycol ..... 913
	\@fllsetnum: Rogue space removed .. 961
	\@specialoutput: Cut-off point changed to 2\baselineskip .... 916
	Empty column action added: \@emptycol ..... 916
	Extra empty column added for twocolumn case ..... 916
	Extra empty column added for twocolumn case (wrong, see below) ..... 916
	\@topnewpage: Added setting of \col@number ..... 913, 914
	Cut-off point changed to 3\baselineskip ..... 915
	Empty column action added: \@emptycol ..... 915
	Message changed for Frank ..... 915
	General: \@activechar@warning changed to an info message. .... 898
	Added \col@number. .... 898
	Documentation tidied. .... 898
	Empty column action added. .... 898
	Fixed bug from \dblfigrule with \@topnewpage. .... 898
	Full of floats action improved. .... 898
	\col@number: Added \col@number .. 910
	\onecolumn: Added setting of \col@number ..... 912
1994-05-01 lterror.dtx v1.0k	\@lateinfo: (CAR) Added draft \@lateinfo. ..... 285
1994-05-01 ltoutenc.dtx 1.4a	General: Added the \a command. ... 363
	Added the \SaveAtCatcode and \RestoreAtCatcode commands. .. 367
	Removed the uc/lc table settings, since the T1 uc/lc table is now the default. .... 375
	Rewrote for the new syntax. .. 367, 369
1994-05-01 ltoutenc.dtx v1.4a	General: Removed Rokicki's encoding. .... 351
	Renamed the commands, removed the \EncodingSpecific command.

Turned all slots into decimal.	1994-05-03 ltmiscen.dtx v1.0h
Added \a. .... 351	\@centercr: \@badcrerr replaced by \@nolnerr ..... 626
1994-05-02 ltcntrl.dtx v1.0l	1994-05-03 ltab.dtx v1.0d
\@break@tfor: Macro added (from ltfiles.dtx) ..... 278	\@endpbox: Use \@finalstrut based on depth of \@arstrutbox ..... 705
1994-05-02 ltdefns.dtx v1.1f	1994-05-04 ltcncls.dtx v1.0b
\@renewcommand: Removed surplus \space in error ..... 84	\NeedsTeXFormat: Changed wording of the warning ..... 813
\@renewenvironment: Removed surplus \space in error ..... 85	1994-05-04 lterror.dtx v1.0m
1994-05-02 ltfiles.dtx v1.0f	\@badcrerr: Error message removed ..... 288
\@iffilenonpath: \@break@loop renamed to \@break@tfor ..... 345	1994-05-05 ltbibl.dtx v1.0c
\@obsoletefile: Make \@onlypreamble ..... 349	\@citex: Set switch for warning and end of run. ..... 784
1994-05-02 ltfinal.dtx v0.1e	\@nocite: Do not write page number in \@nocite warning message. ..... 785
General: Added setting the ‘letter’ catcodes. ..... 986	Set switch for warning and end of run. ..... 785
Added setting the ‘other’ catcodes. 986	1994-05-05 ltfinal.dtx v0.1g
Added setting the special catcodes. 986	General: Added empty errhelp. .... 974
Made slot 127 illegal ..... 986	\@errhelp: Set error help empty. .... 989
Set all the catcodes ..... 974	1994-05-05 ltnntcmd.dtx v3.3c
1994-05-02 ltfinal.dtx v0.1f	\@math@egroup: Corrected \@fontswitch and added saved versions ..... 574
General: Set the catcode of control-J. 986	General: Corrected \@fontswitch .. 567
1994-05-02 ltmiscen.dtx v1.0g	1994-05-05 ltmiscen.dtx v1.0i
General: Changed 91 to 1991 and moved some bits ..... 614	General: Removed braces from ifnextchar and ifstar arguments .. 614
1994-05-02 ltoutput.dtx v1.0o	1994-05-07 ltab.dtx v1.0c
\@resetfps: Code shortened ..... 961	\@maxtab: Changed \@firsttab to \chardef ..... 686
General: Code of \@resetfps shortened. ..... 898	Changed \@maxtab to \chardef .. 686
1994-05-03 ltbibl.dtx v1.0b	General: Removed definition of \+ .. 682
\@nocite: Make \@nocite issue a warning for an undefined citation key. ..... 785	Removed surplus braces from \@ifnextchar constructs ..... 682
1994-05-03 ltfinal.dtx v0.1f	1994-05-08 ltnntcmd.dtx v3.3d
General: Set the catcode of control-J to be ‘other’, for use in messages. 974	General: Removed \@undefinedfonterror ..... 567
1994-05-03 ltfloat.dtx v1.0f	\@normalsize: Removed \@undefinedfonterror ..... 575
General: (CAR) Added \@largefloatcheck ..... 756	1994-05-09 ltnntcmd.dtx v3.3f
Removed unnecessary braces from arguments of \@ifnextchar ..... 756	General: Replaced all \next by \@let@token and undo change 3.3e, whatever that was. .... 567
\@end@dblfloat: \@largefloatcheck added ..... 764	1994-05-10 ltdefns.dtx v1.0n
\@end@float: (CAR) Added \@largefloatcheck ..... 763	General: (ASAJ) Added \DeclareProtectedCommand. .... 78
1994-05-03 ltfsdcl.dtx v2.1f	Added \DeclareProtectedCommand .. 87
General: Renamed \@DeclarerMathDelimiter to \@DeclarerMathDelimiter ..... 485	Removed braces around \@ifundefined argument. ASAJ. 84
1994-05-03 ltlists.dtx v1.0d	\@makeatother: Added \makeatletter and \makeatother ASAJ. .... 105
\@item: \hskip changed to \kern .. 662	
\@item: Removed superfluous braces 661	

1994-05-10 lterror.dtx v1.0n	\@latexerr: (ASAJ) Added extra blank lines to \@latexerr. . . . .	285	\DeclareTextAccent: Reimplemented using \DeclareTextCommand. . . . .	357	
1994-05-10 ltmiscen.dtx v1.0j	\@@sverb: Slight change in error message text. . . . .	632	\hspace: Use \DeclareRobustCommand. ASAJ. . . . .	326	
1994-05-11 ltboxes.dtx v1.0f	\@begin@tempboxa: Use new \color@setgroup concept. . . . .	668	\finalstrut: macro added . . . . .	681	
	\@iiiminipage: Use new \color@setgroup concept. . . . .	677	\fbox: New definition, merged with \framebox . . . . .	672	
	\@mpfootnotetext: Use new \color@setgroup concept. . . . .	678	\framebox: Merged \fbox and \framebox . . . . .	673	
	Use new \normalcolor and \finalstrut. . . . .	678	\normalcolor: macro added for color support . . . . .	670	
General: Superfluous braces removed from several commands . . . . .	667	1994-05-12 ltdefns.dtx v1.0p	General: (ASAJ) Fixed a bug with \relax which was using \@gobble before defining it. . . . .	78	
\color@setgroup: macro added for color support . . . . .	670		Fixed a bug with \relax which was using \@gobble before defining it. . . . .	87	
\endminipage: Use new \color@setgroup concept. . . . .	677	1994-05-12 ltfssbas.dtx v2.1j	General: New baselinestretch concept Replaced hand-protected commands by \DeclareRobustCommand defs . . . . .	407	
1994-05-11 ltclass.dtx v1.0c	\endfilecontents: Add checks for form feed and tab . . . . .	824	\f@linespread: New macro . . . . .	417	
1994-05-11 ltdirchk.dtx v1.0e	General: Add \ProvidesFile as used in fd files. . . . .	4	\fontencoding: Use \DeclareRobustCommand. . . . .	415	
1994-05-11 lterror.dtx v1.0o	\@latexerr: (ASAJ) Removed one of the extra blank lines to \@latexerr. . . . .	285	\fontfamily: Use \DeclareRobustCommand. . . . .	416	
1994-05-11 ltlogos.dtx v1.0o	\LaTeX: Use \DeclareProtectedCommand. ASAJ. . . . .	329	\fontseries: Use \DeclareRobustCommand. . . . .	416	
	\LaTeXe: Use \DeclareProtectedCommand. ASAJ. . . . .	329	\fontshape: Use \DeclareRobustCommand. . . . .	416	
1994-05-11 ltoutenc.dtx 1.5a	General: Made T1 and OT1 generate packages rather than def files. Renamed the ‘package’ module to ‘teststy’. . . . .	354	\fontsize: Redefined to use \set@fontsize . . . . .	417	
1994-05-11 ltoutenc.dtx v1.5a	General: Reimplemented \DeclareTextCommand using \@changed@cmd and \DeclareProtectedCommand. . . . .	355	\linespread: New macro . . . . .	417	
	Renamed the commands again. Made the encoding part of the command syntax. Added the \DeclareTextCommand interface. Used \DeclareProtectedCommand. . . . .	351	\mathversion: Use \DeclareRobustCommand. . . . .	418	
1994-05-12 ltfssdcl.dtx v2.1g	General: Allow \relax as undefined command . . . . .	485	1994-05-12 ltfssini.dtx v2.1i	General: Allow \relax’ed cmds to be declared . . . . .	485
	Allow \relax’ed cmds to be declared . . . . .	485		General: Moved \fontencoding to fam.dtx . . . . .	515
1994-05-12 ltfssini.dtx v2.1i	Moved \fontfamily to fam.dtx . . . . .	515	Moved \fontseries to fam.dtx . . . . .	515	
	Moved \fontshape to fam.dtx . . . . .	515	Moved \fontsize to fam.dtx . . . . .	515	
	Moved \mathversion to fam.dtx . . . . .	515	Moved \selectfont to tracefnt.dtx	515	

1994-05-12 lfsstrc.dtx v2.3f	1994-05-13 ltfinal.dtx v1.0h
\selectfont: Use \DeclareRobustCommand . . . . . 459	General: Added output enc stuff . . . . . 988
1994-05-12 ltoutenc.dtx v1.5a	1994-05-13 ltfloat.dtx v1.0g
General: Removed the \SaveAtCatcode and \RestoreAtCatcode commands. . . . . 367	\@footnotetext: (DPC) Add new style colour support: \normalcolor . . . . . 775
Rewrote for the new syntax. . . . . 367, 369	(DPC) Use \finalstrut . . . . . 775
1994-05-12 ltoutput.dtx v1.0p	\@xfloat: (DPC) Use \normalcolor . . . . . 761
\@writeshadow: \normalcolor added . . . . . 925	1994-05-13 lfntcmd.dtx v3.3g
General: \normalcolor added in various places (DPC). . . . . 898	General: Replaced \protecteddef by \DeclareRobustCommand . . . . . 567
1994-05-13 ltboxes.dtx v1.0h	1994-05-13 lfssbas.dtx v2.1k
\@arrayparboxrestore: New accent system, use \let not \def . . . . . 676	General: Remove File identification 'typeout' . . . . . 407
1994-05-13 ltcnts.dtx v1.0f	1994-05-13 lfssbas.dtx v2.1l
General: Removed \Ialpha . . . . . 402	\DeclareFontEncoding: Init encoding change command . . . . . 411
Removed \ialpha . . . . . 402	\define@newfont: Use \input@ for fd files . . . . . 421
1994-05-13 ltdefns.dtx v1.0q	1994-05-13 lfssdcl.dtx v2.1h
General: (ASAJ) Renamed \DeclareProtectedCommand to \DeclareRobustCommand.	General: Removed file identification typeout . . . . . 485
Removed \if@short@command. . . . . 78	1994-05-13 lfssini.dtx v2.1j
(ASAJ) Replaces \space by ' ' in \csname. . . . . 78	General: Removed file identification typeout . . . . . 515
Renamed \DeclareProtectedCommand to \DeclareRobustCommand.	1994-05-13 lfsstrc.dtx v2.3g
Removed \if@short@command.	General: Removed timeouts as \ProvidesPackage writes to log. . . . . 455
Moved to after the definition of \@gobble. . . . . 87	1994-05-13 ltoutenc.dtx v1.5b
1994-05-13 ltdefns.dtx v1.0r	General: Added \{, \} and \\$. . . . . 351
General: (ASAJ) Added logging message to \DeclareProtectedCommand. . . . . 78	Renamed \DeclareProtectedCommand to \DeclareRobustCommand. . . . . 351
Added logging message to \DeclareProtectedCommand. . . . . 87	Replaces \space by ' ' in \csname. . . . . 351
1994-05-13 ltdefns.dtx v1.0s	1994-05-13 ltpictur.dtx v0.1d
General: (ASAJ) Added \backslashchar. . . . . 78	General: Removed surplus braces from \if.. constructions . . . . . 707
(ASAJ) Coded \ifdefinable more efficiently. . . . . 78	1994-05-13 ltab.dtx v1.0d
Coded more efficiently, thanks to FMi. . . . . 84	\@contfield: Colour support . . . . . 688
1994-05-13 ltfiles.dtx LaTeXe	\@startfield: Colour support . . . . . 688
\listfiles: Stop \listfiles being run twice . . . . . 349	\@stopfield: Colour support . . . . . 688
1994-05-13 ltfiles.dtx v1.0g	\@a: moved to ltoutenc . . . . . 686
\document: Added execution of \every@size . . . . . 333	1994-05-14 fontdef.dtx v2.1f
1994-05-13 ltfinal.dtx v0.1h	General: Removed .def files. . . . . 545
General: Added package otlenc, and defined \acci, \accii and \acciii. . . . . 974	1994-05-14 lfssbas.dtx v2.1m
	\enc@update: Macro added . . . . . 416
	1994-05-14 lfssbas.dtx v2.1n
	General: Set defaults for all \f@... . . . . . 417
	\DeclareErrorFont: Don't set \f@encoding . . . . . 424
	\DeclareFontEncoding: Log if encoding is redeclared . . . . . 411
	Only init enc change cmd when new encoding . . . . . 411

1994-05-14 ltfssini.dtx v2.1k		1994-05-16 ltlogos.dtx v1.1a	
General: Init error font just before checking for fontdef.cfg . . . . .	540	General: (ASAJ) Split from ltinit.dtx.	329
\reset@font: Remove surplus braces	538	1994-05-16 ltmath.dtx v1.0k	
1994-05-14 ltfsstrc.dtx v2.3h		\ensuremath: Use \DeclareRobustCommand and add extra braces in math mode . . . . .	646
\selectfont: Added \enc@update . . . . .	461	1994-05-16 ltoutenc.dtx 1.5h	
1994-05-14 ltoutenc.dtx 1.5d		General: \pounds was still using u rather than ui shape. . . . .	367
General: Moved the driver to the top. . . . .	354	1994-05-16 ltoutenc.dtx v1.5f	
1994-05-14 ltoutenc.dtx v1.5c		General: enc files now have uc encoding name parts (FMi) . . . . .	351
General: Added the fontenc package . . . . .	394	Revert code so that the encoding given is used in \DeclareTextCommand (FMi) . . . . .	351
Added the fontenc package. . . . .	351	1994-05-16 ltoutenc.dtx v1.5g	
Fixed a bug which caused an infinite loop if \f@encoding was incorrectly set. . . . .	351, 355	General: Made fontenc.sty use the new mixed-case encoding files. . . . .	351
Moved fontsmp to its own dtx file. . . . .	351	Removed the lowercasing of the filename. . . . .	394
1994-05-14 ltoutenc.dtx v1.5d		1994-05-16 ltoutenc.dtx v1.5h	
General: Rewrote \DeclareTextCommand to define its argument to use the current encoding by default, rather than the encoding provided to \DeclareTextCommand. . . . .	351, 355	General: Added \NG, \ng, \TH, \th, \DH, \dh, \DJ and \dj. . . . .	351
Tidied up the documentation. . . . .	351	Added \r (ring accent) and \k (ogonek) accents. . . . .	351
1994-05-14 ltoutenc.dtx v1.5e		Fixed a bug with \pounds. . . . .	351
General: Replaced \ENC@cmd by \ENC-cmd. . . . .	351	Removed \P from the OT1 definitions file. . . . .	351
1994-05-15 ltfsbas.dtx v2.1o		1994-05-16 ltoutenc.dtx v1.5i	
General: encoding cmd changed to enc-cmd . . . . .	407	General: Fixed a bug with \d. . . . .	351
1994-05-16 fontdef.dtx v2.1g		1994-05-16 ltoutput.dtx v1.0q	
General: Removed \DeclareFontEncoding for ot1 and t1 and input .def files instead . . .	545	\@writetext: Changed setting of accents (FMi): with the new encoding setup they can use \let. It could also use the new internal commands? . . . . .	926
1994-05-16 ltalloc.dtx v1.1a		General: Changed setting of accents (FMi). . . . .	898
General: (ASAJ) Split from ltinit.dtx. . . . .	273	1994-05-16 ltpar.dtx v1.1a	
1994-05-16 ltcntrl.dtx v1.0a		General: (ASAJ) Split from ltinit.dtx.	290
General: (ASAJ) Split from ltinit.dtx. . . . .	275	1994-05-16 lplain.dtx v1.0h	
1994-05-16 ltfdfns.dtx v1.1a		General: Comment out encoding specific commands . . . . .	33
General: (ASAJ) Split from ltinit.dtx. . . . .	78	Remove \@acci and friends again .	34
1994-05-16 lterror.dtx v1.1a		Remove unnecessary def for \item .	33
General: (ASAJ) Completely new error interface. . . . .	279	\loop: Use Kabelschacht method . . .	31
(ASAJ) Split from ltinit.dtx. . . . .	279	\m@th: Remove unnecessary space . . .	32
1994-05-16 ltfinal.dtx v1.0i		1994-05-16 ltspace.dtx v1.1a	
General: moved output enc stuff to lfonts . . . . .	988	General: (ASAJ) Split from ltinit.dtx.	310
1994-05-16 ltfsbas.dtx v2.1p		1994-05-17 ltclass.dtx v1.0e	
\fontsize: Pass \baselinestretch not \f@linespread . . . . .	417	\use@option: Execute option after removing from list, not before . . .	810
\linespread: Remove surplus braces	417		
1994-05-16 ltfssini.dtx v2.1m			
\@acci: Define saved versions of accents . . . . .	541		

1994-05-17 ltdefns.dtx 1.1b	General: (ASAJ) Added the <code>\@protect@...</code> commands. . . . .	88	1994-05-19 ltcntlen.dtx v1.1a	General: Extracted file from ltcntlen. . . . .	397
1994-05-17 ltdefns.dtx v1.1b	General: (ASAJ) Added definitions for protect. . . . .	78	1994-05-19 ltdefns.dtx v1.1d	General: (RmS) Added definitions for <code>\@namedef</code> and <code>\@nameuse</code> again. . . . .	78
	(ASAJ) Removed warnings and logging to lterror.dtx. . . . .	78	1994-05-19 ltfinal.dtx v0.1k	General: Removed <code>\makeat...</code> . . . . .	974
	Added the discussion of protected commands, defined the values that <code>\protect</code> should have. . . . .	88	1994-05-19 ltidxglo.dtx v1.1a	General: Initial version of ltidxglo.dtx, split from ltidxbib.dtx . . . . .	779
1994-05-17 ltdefns.dtx v1.1c	General: (ASAJ) Redid definitions for protect. . . . .	78	1994-05-19 ltlenth.dtx v1.1a	General: Extract file ltlenth from ltcntlen. . . . .	405
1994-05-17 lterror.dtx v1.1b	General: (ASAJ) Moved error stuff from ltdefns.dtx. . . . .	279	1994-05-19 ltpageno.dtx v1.1a	General: Extract file ltpageno from ltcntlen. . . . .	608
1994-05-17 ltfsmini.dtx v2.1n	<code>\copyright</code> : Really add extra braces <code>\nfss@text</code> : Added braces to allow use in subscripts . . . . .	538	1994-05-19 lplain.dtx v0.1k ltfinal	<code>\showoutput</code> : used <code>\maxdimen</code> not 99999 . . . . .	34
				<code>\showoverfull</code> : used <code>\one</code> not 1 . . . . .	34
1994-05-17 ltmath.dtx v1.0i	General: Replaced <code>\let</code> by <code>\gdef</code> , for indirect definition. . . . .	642	1994-05-19 ltxref.dtx v1.1a	General: Extract file ltxref from ltcntlen. . . . .	609
1994-05-17 ltoutenc.dtx v1.5j	General: Added braces to <code>\pounds</code> so it works as a subscript. . . . .	351	1994-05-20 ltdefns.dtx v1.1e	General: Changed command name from <code>\@checkcommand</code> to <code>\CheckCommand</code> . . . . .	78
1994-05-18 ltdefns.dtx 1.1c	General: (ASAJ) Renamed the commands, and removed one which is no longer needed. . . . .	88		<code>\CheckCommand</code> : Changed name from <code>\@checkcommand</code> to <code>\CheckCommand</code> . . . . .	86
1994-05-18 ltdefns.dtx v1.1c	General: Redid the discussion and definitions, in line with the proposed new setting of <code>\protect</code> in the output routine. . . . .	88	1994-05-20 lterror.dtx v1.1c	General: (ASAJ) Added <code>\@later@info@no@line</code> . . . . .	279
				(ASAJ) Added missing full stops. . . . .	279
1994-05-18 ltfinal.dtx v0.1j	General: Corrected the lccode for d-bar. . . . .	974		(ASAJ) Fixed a bug with <code>\@inmatherr</code> . . . . .	279
1994-05-18 ltlogos.dtx v1.1b	General: (ASAJ) Added the L <sup>E</sup> T <sub>E</sub> X logo. . . . .	329	1994-05-20 ltfinal.dtx v0.1l	General: Use new font warning commands . . . . .	981
	(ASAJ) Made the L <sup>E</sup> T <sub>E</sub> X 2 <sub>E</sub> logo use the text font '2' rather than the math font '2'. . . . .	329	1994-05-20 ltfloat.dtx v1.0h	<code>\@endfloatbox</code> : Restore outer value of @nobreak switch. . . . .	765
1994-05-18 ltoutenc.dtx v1.5k	General: Made dotted-i produce 'i'. . . . .	351		<code>\outer@nobreak</code> : Macro added: default is to do nothing. . . . .	765
	Removed braces from <code>\pounds</code> and <code>\dollar</code> . . . . .	351	1994-05-20 ltfnctcmd.dtx v3.3h	General: Use new error commands . . . . .	567
	Replaced <code>\defaultencoding</code> with <code>\encodingdefault</code> . . . . .	351	1994-05-20 ltfsbas.dtx v2.1q	General: Use new error commands . . . . .	407
1994-05-19 ltbibl.dtx v1.1a	General: Initial version of ltbibl.dtx, split from ltidxbib.dtx . . . . .	782	1994-05-20 ltfsstrc.dtx v2.3i	General: Use new error command names . . . . .	455
			1994-05-20 ltmiscen.dtx v1.0l	<code>\@writefile</code> : Added correct setting of <code>\protect</code> . . . . .	620

1994-05-20 ltmiscen.dtx v1.0m	\GenericError, \GenericWarning and \GenericInfo. ....	279
General: Use new warning commands		614
1994-05-20 ltoutput.dtx v1.0s	(ASAJ) Replaces \string by \protect in some messages. . .	279
\@writesetup: Added setting of \protect during \shipout. ....	925	
General: Added setting of \protect during \shipout. ....	898	
1994-05-20 ltpage.dtx v1.0d		
\markright: Changed setting for \protect. ....	788	
1994-05-20 ltsect.dtx v1.0c		
General: Correct setting of \protect.	753	
\addcontentsline: Correct setting of \protect. ....	752	
1994-05-21 ltbibl.dtx v1.1b		
General: Use new warning commands	782	
1994-05-21 lterror.dtx v1.1d		
General: (ASAJ) Made the error commands robust. ....	279	
1994-05-21 ltfiles.dtx v1.0h		
General: Use new error commands	330	
1994-05-21 ltlists.dtx v1.0f		
General: Use new error commands	650	
1994-05-21 ltmiscen.dtx v1.0n		
General: Use new error commands	614	
1994-05-21 ltsect.dtx v1.0d		
General: Use new error commands	743	
1994-05-21 ltab.dtx v1.0f		
General: Use new error commands	682	
1994-05-21 ltxref.dtx v1.1b		
General: Use new warning commands	609	
\newlabel: Use new warning commands ....	611	
1994-05-22 ltclass.dtx v1.0f		
General: Use new warning and error commands ....	791	
1994-05-22 ltdefns.dtx v1.1f		
General: Use new warning and error cmds ....	78	
1994-05-22 lterror.dtx v1.1e		
General: (ASAJ) Replaced bgroup by begingroup in error messages, to stop extra mathords creeping into math mode. ....	279	
1994-05-22 lterror.dtx v1.2a		
General: (ASAJ) Made \GenericError, \GenericWarning and \GenericInfo robust. ....	279	
(ASAJ) Replaced \\ and tilde by \MessageBreak and \space. ....	279	
(ASAJ) Replaced \@generic@message and \@generic@error by		
1994-05-22 ltfloat.dtx v1.0i	\GenericError: (DPC) Alternative version added for old TeXs . .	280
1994-05-22 ltoutput.dtx v1.0t	(DPC) New version using long command name. ....	280
1994-05-22 ltpictur.dtx v0.1e		
General: Use new warning cmd ..	707	
1994-05-23 ltclass.dtx v1.0h		
\NeedsTeXFormat: Don't stop completely when format is wrong	813	
\usepackage: Remove argument if possible ....	813	
1994-05-23 ltdirchk.dtx v1.0f		
General: Document \CTeXversion . .	1	
1994-05-23 ltfsstrc.dtx v2.3j		
General: Removed def of \f@warn@break ....	473	
1994-05-23 ltoutput.dtx v1.0u	\@activechar@info: Added \MessageBreak ....	925
\@writesetup: Changed resetting of \protect after shipout to use \aftergroup ....	925	
General: Added \MessageBreak. ....	898	
	Changed resetting of \protect after shipout. ....	898
1994-05-24 lterror.dtx v1.2e		
\@latex@info@no@line: Macro added	283	
1994-05-24 lterror.dtx v1.2f		
General: (DPC) wrap long lines ..	279	
1994-05-24 ltntcmd.dtx v3.3i		
General: Tidying and typos fixed ..	567	
1994-05-24 ltmiscen.dtx v1.0q	\@currenvline: Use \empty as outer default ....	625
1994-05-25 ltdirchk.dtx v1.0g	\filename@parse: Mac parser had " typo for : ....	12
1994-05-25 ltntcmd.dtx v3.3j		
General: Insertion of \aftergroups to implement \nocorr moved to the end of the group .....	567	
\check@icr: Macros added .....	571	

\check@nocorr@: Insertion of \aftergroups moved and defaults set up for efficiency . . . . .	571	1994-06-08 ltfinal.dtx v1.0m General: Add patch file system . . . . .	988
\DeclareTextFontCommand: \expandafter inserted . . . . .	569	1994-06-09 ltfinal.dtx v1.0n General: For TeX2, do not set codes for higher half of character table. . . . .	979, 986
Insertion of \aftergroups moved . . . . .	569	1994-06-09 ltfntcmd.dtx v3.3k General: Tidying and typos fixed in documentation . . . . .	567
1994-05-25 ltoutput.dtx v1.0v General: Extra documentation. . . . .	898	1994-06-18 ltfntcmd.dtx v3.3l General: Added check for empty text	567
1994-05-25 ltsect.dtx v1.0e \@dottedtocline: Put braces around argument 4 (the actual toc entry) to avoid font (and possibly other) changes leaking out to the leaders.	754	\check@nocorr@: Added check for empty text . . . . .	571
1994-05-25 ltthm.dtx v1.0c General: Modify documentation . . .	739	1994-06-22 ltfntcmd.dtx v3.3m General: Removed space from \nfss@text . . . . .	567
1994-05-25 ltvers.dtx v1.0d General: Remove PRELIMINARY TEST RELEASE from startup banner (spring is here) . . . . .	39	Renamed \check@nocorr . . . . .	567
1994-05-25 ltxref.dtx v1.1c General: Modify documentation . . .	609	\check@nocorr@: Renamed \check@nocorr to \text@command to improve \long error message .	571
1994-05-26 ltfiles.dtx LaTeX2e \@missingfileerror: Modify message format . . . . .	347	\DeclareTextFontCommand: Removed space from \nfss@text . . . . .	569
1994-05-26 ltlogos.dtx v1.1c General: Remove \SLiTeX logo . . .	329	1994-06-22 ltmath.dtx v1.2t classes \mathindent: Set \mathindent at the end of the class instead of at begin document . . . . .	647
1994-05-26 ltmiscen.dtx v1.0r General: \literal removed . . . . .	634	1994-07-20 ltlogos.dtx v1.1e \LaTeX: Save a few tokens . . . . .	329
1994-05-26 ltplain.dtx v1.1m \iterate: (CAR) added \long . . . . .	31	\LaTeXe: Save a few tokens . . . . .	329
\underbar: (CAR/FMi) changed to use box \tw@ . . . . .	32	1994-07-20 ltpage.dtx v1.0h \sloppy: Save a few tokens . . . . .	789
1994-05-26 ltplain.dtx v1.1p \underbar: (DPC) changed to use \sbox . . . . .	32	1994-09-16 ltfssbas.dtx v2.1s \nfss@catcodes: Reset [ and ] as well, just in case . . . . .	422
1994-05-29 ltfssdcl.dtx v2.1j General: Use new error commands .	485	1994-10-07 ltoutenc.dtx v1.5l General: Moved the ogonek accent. .	351
1994-05-31 ltfinal.dtx v1.0n General: Renamed lthyphen.* to lthyphen.*. . . . .	974	1994-10-11 ltdirchk.dtx v1.0h \@TeXversion: Check for TeX3.14 . .	13
1994-06-01 ltboxes.dtx v1.0i \@frameb@x: Macro added. . . . .	673	General: Modify all of ltxcheck again .	14
\@iframebox: New version, so \width is correct in \framebox . . . . .	673	1994-10-12 ltsect.dtx v1.0f General: Doc. typos . . . . .	743
\fbox: New version, using \@frameb@x . . . . .	672	1994-10-14 fontdef.dtx v2.2a General: New coding . . . . .	543
\framebox: New version, so \width is correct in \framebox . . . . .	673	1994-10-14 ltfssini.dtx v2.2a General: New coding for cfg files . .	515
1994-06-01 ltlogos.dtx v1.1d \LaTeX: Add \m@th to force math size calculations . . . . .	329	1994-10-14 ltmiscen.dtx v1.0s General: Move math to other file . .	614
1994-06-01 ltoutput.dtx v1.0w General: Tidied up typesetting. . .	898	1994-10-14 ltplain.dtx v1.1a General: Moved code to other files. . .	15
		1994-10-15 ltfssbas.dtx v2.1t \extract@alpha@from@version: Warn if math alpha is used outside math . . . . .	429

1994-10-18 ltboxes.dtx v1.0j		1994-10-25 ltdefns.dtx v1.2b
\@frameb@x: \leavevmode added . . .	673	General: Documentation
\@iframebox: \leavevmode moved to \@frameb@x . . . . .	673	improvements . . . . .
\@parboxto: Macro added to remove misuse of \empty . . . . .	675	78
General: stuff from ltpatch done . . .	667	
\fbox: \long added . . . . .	672	
\mbox: \long added . . . . .	668	
\sbox: \long added . . . . .	671	
1994-10-18 ltclass.dtx v1.0j		1994-10-25 ltoutenc.dtx 1.6a
General: Move \listfiles to ltfiles.dtx . . . . .	791	General: Added \textdollar, \textlbrace, \textrbrace, \textsterling, \textunderline. . . . .
1994-10-18 ltdefns.dtx v1.2a		369
\@star@or@long: macro added . . . . .	81	Removed \textlbrace, \textrbrace, \textunderline to give them their proper names. . . . .
General: Add extra test for \endgraf . . .	78	369
Add star-forms for all commands . . . . .	78	
\renewenvironment: reset end command . . . . .	85	1994-10-25 ltoutenc.dtx v1.6a
1994-10-18 ltfiles.dtx v1.0i		General: Added
\listfiles: code moved here from ltclass . . . . .	349	\ProvideTextCommand, \UseTextSymbol, \UseTextAccent, \DeclareTextSymbolDefault, \DeclareTextAccentDefault, \DeclareTextCommandDefault, and
1994-10-18 ltoutenc.dtx v1.5l		\ProvideTextCommandDefault. . . . .
General: Added new definitions of \patterns and \hyphenation. . . . .	363	351
1994-10-18 ltoutenc.dtx v1.5m		Added the \Provide commands, and the default definitions. . . . .
General: Added new definitions of \patterns and \hyphenation. . . . .	351	355
1994-10-18 ltsect.dtx v1.0g		Added the defaults. . . . .
\@dottedtocline: Added \normalcolor for page number . . .	754	363
General: Added \normalcolor . . . . .	743	Added the files OT1enc.def, T1enc.def and OMSenc.def. . . . .
1994-10-19 ltfssbas.dtx v2.1t		363
\DeclareFontEncoding: Add missing \relax. . . . .	411	Added the OMS encoding. . . . .
1994-10-23 lfsstrc.dtx v23.k		375
\every@math@size: Renamed to \every@math@size . . . . .	464	1994-10-27 ltoutenc.dtx 1.6b
1994-10-23 ltmath.dtx v1.0l		General: Added \textasciicircum \textasciitilde \textbackslash \textbar \textbraceleft \textbraceright \textcompwordmark \textemdash \textendash \textexclamdown \textgreater \texthyphenchar \texthyphen \textless \textquestiondown \textquotedblleft \textquotedblright \textquotedbl \textquotelleft \textquoteright \textunderscore \textvisiblespace . . . . .
\@eqnnum: Added \normalcolor since \eqno introduces a subgroup of the displayed math group . . . . .	644	369
\ensuremath: Remove extra braces: but see p 168 of Leslie's book . . .	646	Added: \textemdash \textendash \textexclamdown \texthyphenchar \texthyphen \textquestiondown \textquotedblleft \textquotedblright \textquotelleft \textquoteright . . . . .
1994-10-24 ltboxes.dtx v1.0k		367
\fbox: Inner braces added (to fix latex/1061) . . . . .	672	1994-10-27 ltoutenc.dtx v1.5d
1994-10-25 fontdef.dtx v2.2c		General: Rewrote
General: Added OMSenc.def . . . . .	545	\DeclareTextSymbol to define its argument to use the current
1994-10-25 ltboxes.dtx v1.0l		
\@isavepicbox: missing percent (moved from ltpatch) . . . . .	671	

encoding by default, to fit with \DeclareTextCommand. . . . .	355	Rewrote \copyright to use \textcircled. . . . .	365
1994-10-27 ltoutenc.dtx v1.6b		1994-10-31 fontdef.dtx v2.2d	
General: Added \textbackslash. . . . .	375	General: Added OMLenc.def ... . . .	545
Added more defaults for OT1. . . . .	363	1994-10-31 fontdef.dtx v2.2e	
Removed the enc.def files . . . . .	351	General: ... and moved further down	545
Removed the files OT1enc.def, T1enc.def and OMSenc.def. . . . .	363	1994-10-31 ltfloat.dtx v1.1a	
Renamed \textlbrace to \textbraceleft and \textrbrace to \textbraceright. . . . .	375	\@dblfloat: Major changes since two-column and one-column cases merged . . . . .	759
1994-10-29 ltmath.dtx 1.0m		\@dblflset: Macro added . . . . .	759
General: ASAJ: Added \DeclareMathOperator. . . . .	635	Major changes to parameter parsing, setting of local variables, etc; two-column and one-column cases merged; space hacks moved	759
ASAJ: Tidied up documentation.	642	\@endfloatbox: (DPC/CAR) Extra box added to remove colour resetting from vmode . . . . .	765
1994-10-29 ltmath.dtx v1.0m		\@floatboxreset: Macro added . . . . .	763
General: ASAJ: Added \mathellipsis, \mathdollar and \mathsterling. . . . .	642	\@footnotetext: (DPC/CAR) Move colour setting to output routine .	775
ASAJ: Removed \dag, \ddag. . . . .	642	\@savemarbox: (DPC/CAR) Extra box added for colour . . . . .	769
ASAJ: Renamed \S and \P to \mathsection and \mathparagraph and made them \mathchardef. . . . .	642	\@setfps: Macro added . . . . .	760
1994-10-29 ltoutenc.dtx v1.6c		\@xdblfloat: Macros removed: \@dbflt, \@xdblfloat . . . . .	765
General: Added commands like \dots for use in text and math. . . . .	363	\@xfloat: (DPC/CAR) Extra box added to remove colour resetting from vmode . . . . .	761
Renamed \P, \S, \dag and \ddag to \textparagraph, \textsection, \textdagger and \textdaggerdbl. . . . .	351	Major changes, removing setting of local variables, space hacks etc; two-column and one-column cases merged . . . . .	760
1994-10-30 ltdefns.dtx v1.2c		Reset hook added . . . . .	761
\@onelvel@sanitize: Macro added	104	\@xmpar: (DPC/CAR) Extra box added since needed for floats . . .	770
General: (CAR)\@onelvel@sanitize added . . . . .	78	\fps@dbl: Macro added . . . . .	760
1994-10-30 ltdefns.dtx v1.2f		1994-10-31 ltoutput.dtx v1.1a	
General: (DPC)\newwrite's moved to ltfiles . . . . .	78	\@makecol: (DPC/CAR) Colour resetting moved to here . . . . .	922
1994-10-30 ltmath.dtx v1.0n		\@topnewpage: (DPC/CAR) Extra box added to remove colour resetting from vmode . . . . .	914
General: ASAJ: Moved the new commands to ltoutenc. . . . .	642	(DPC/CAR) Use \color@begingroup for colour . . .	914
1994-10-30 ltoutenc.dtx v1.6d		(DPC/CAR) Use \normalcolor .	914
General: Added \DeclareTextCompositeCommand. . . . .	351	1994-11-02 ltoutenc.dtx v1.6d	
Added \textcircled. . . . .	351, 365, 375	General: Wrapped lines longer than 70 characters. . . . .	351
Added \t. . . . .	365	1994-11-03 ltclass.dtx v1.0k	
Added math commands. . . . .	351	General: Move \@missingfileerror to ltfiles . . . . .	795
Added OML encoding. . . . .	351, 365		
Added the OML encoding. . . . .	376		
Made \textless and \textgreater come from OML. . . . .	365		
Moved math commands here from ltmath. . . . .	367		
Removed \textregistered. . . . .	365		

1994-11-03 ltdirchk.dtx v1.0i		1994-11-04 ltsect.dtx 1.0h	
General: Generate an error if latex.ltx not used with clean initex . . . . .	1	\@sect: (ASAJ) Added \protected@edef. . . . .	747
1994-11-03 ltfiles.dtx v1.0j		General: (ASAJ) Added \protected@xdef to \thanks. . . . .	743
\@missingfileerror: Move here from ltclass . . . . .	347	1994-11-04 ltsect.dtx v1.0h	
1994-11-04 ltboxes.dtx v1.0m		General: Added \protected@write to \addtocontents. ASAJ. . . . .	753
\@mpfootnotetext: Added \protected@edef. ASAJ. . . . .	678	\addcontentsline: Added \protected@write to \addcontentsline. ASAJ. . . . .	752
1994-11-04 ltdefns.dtx v1.2e		1994-11-04 ltab.dtx v1.0h	
General: Added \set@display@protect to \typeout. ASAJ. . . . .	78	\@mkpream: (ASAJ) Added \unexpandable@protect to \@mkpream. . . . .	701
Added commands for setting and restoring \protect. ASAJ. . . . .	90	\multicolumn: (ASAJ) added \set@typeset@protect. . . . .	697
Rewrote protected short commands using \x@protect. ASAJ. . . . .	88	1994-11-04 ltxref.dtx v1.1d	
1994-11-04 lterror.dtx v1.2g		\label: (ASAJ) Added \protected@edef . . . . .	611
General: Added \set@display@protect to \Generic* commands. ASAJ. . . . .	279	(ASAJ) Added \protected@write . . . . .	611
1994-11-04 ltfiles.dtx v1.0k		1994-11-05 ltboxes.dtx v1.0n	
\nofiles: Added setting of \protected@write, \makeindex and \makeglossary to \nofiles. ASAJ. . . . .	337	\@mpfootnotetext: Color resetting for footnotes moved to endminipage: as for main page. . . . .	678
\protected@write: Macro added ASAJ. . . . .	337	\color@endbox: macro added for color support . . . . .	670
1994-11-04 ltfloat.dtx v1.1b		\color@hbox: macro added for color support . . . . .	670
\@footnotetext: (ASAJ) Added \protected@edef. . . . .	775	\endminipage: Color resetting for footnotes moved to here: as for main page. . . . .	677
\footnotemark: Added \protected@xdef to \footnotemark. . . . .	776	1994-11-05 ltboxes.dtx v1.0o	
1994-11-04 ltdxglo.dtx v1.1b		\@mpfootnotetext: Color groups restored here. . . . .	678
\@wrglossary: Added \protected@write to \@wrglossary. . . . .	781	1994-11-05 ltfloat.dtx v1.1c	
\@wrindex: Added \protected@write to \@wrindex. . . . .	780	\@dblflset: Add compatibility with old version of \xfloat. . . . .	759
General: Removed \if@filesw from \makeindex. . . . .	779	\@endfloatbox: Use new \color@hbox concept. . . . .	765
\makeglossary: Removed \if@filesw from \makeglossary. . . . .	780	\@footnotetext: Removed \normalcolor (again) . . . . .	775
1994-11-04 ltmiscen.dtx v1.0t		\@savemarbox: Use new \color@hbox concept. . . . .	769
\@writefile: Removed setting of \protect. ASAJ. . . . .	620	\@setfps: Add compatibility with old version of \xfloat. . . . .	760
1994-11-04 ltoutenc.dtx v1.6f		\@xfloat: Add compatibility with old version of \xfloat: but the arguments, provided at exorbitant cost, are now completely ignored	760
General: Added \_. . . . .	366	Use new \color@hbox concept. . . . .	761
Added \mathunderscore. . . . .	367	\@xmpar: Use new \color@hbox concept. . . . .	770
1994-11-04 ltpage.dtx v1.0e			
\markright: Added \unexpandable@protect. ASAJ.	788		

1994-11-05 ltoutenc.dtx v1.6g		1994-11-10 ltbibl.dtx v1.1c	
General: Added setting of \@typeset@protect to \patterns and \hyphenation. ....	363	General: Fix \nocite{*} ..... \nocite: Fix \nocite{*} .....	782 785
1994-11-05 ltoutput.dtx v1.1b		1994-11-10 ltmath.dtx v1.2v classes	
\@topnewpage: Use new \color@hbox concept. ....	914	\eqnarray: Added value of \parskip to \abovedisplayskip to compensate for negative \topsep ....	649
\@writesetup: Change protect settings for new-style, protect-free aux-files. ....	925	1994-11-10 ltoutput.dtx v1.1e	
Use new \color@hbox concept. ....	925	\@writesetup: Modify \protect setting ....	925
1994-11-05 ltoutput.dtx v1.1c		1994-11-10 lplain.dtx v1.1b	
\@begindvi: Added macro ....	929	General: (CAR) added patch to \loop. 15	
\@begindvibox: Added macro ....	910	\iterate: (CAR) added extra \relax 31	
\@writesetup: Add new \AtBeginDvi concept ....	925	1994-11-11 lspace.dtx v1.2a	
\AtBeginDvi: Added macro ....	910	\\: (DPC) Make robust ....	314
1994-11-06 ltfsbas.dtx v2.1u		1994-11-12 lfnntcmd.dtx v3.3o	
\cf@encoding: New macro ....	417	\normalsize: Added \MessageBreak 575	
\DeclareFixedFont: Renamed \every@size to \every@math@size. ....	409	1994-11-12 llists.dtx v1.2b lspace	
1994-11-06 ltfsini.dtx v2.2b		\endtrivlist: Changed order of tests to make \noitemerror correct: end of an era. ....	659
\@setsize: Use \@typeset@protect 537		1994-11-12 ltmiscen.dtx v1.0u	
1994-11-06 lfsstrc.dtx v2.3k		\center: Changed end macro to \def: safer and consistent ....	627
\glb@currsize: New implementation 463		\flushleft: Changed end macro to \def: safer and consistent ....	628
\try@simples: New implementation 474		\flushright: Changed end macro to \def: safer and consistent ....	628
\try@size@substitution: New implementation ....	474	1994-11-12 lplain.dtx v1.1c	
\tryis@simple: New implementation 474		General: Comment out more encoding specific commands ....	33
1994-11-07 fontdef.dtx v2.2f		1994-11-12 lspace.dtx v1.2b	
General: (DPC) Add \DeclareMathSizes declarations 550		\addpenalty: Corrected error message ....	322
(DPC) Updated to use \ProvidesFile ....	545	\addvspace: Corrected error message 321	
1994-11-07 ltfiles.dtx v1.0m		1994-11-13 lspace.dtx v1.2c	
\document: Renamed \every@size to \every@math@size. ....	333	\addpenalty: Recorrected error message ....	322
1994-11-07 lplain.dtx v1.0l		\addvspace: Recorrected error message ....	321
\@unused: move here from ltdefns, remove duplicate \mainaux ....	24	1994-11-14 ltoutput.dtx v1.1f	
1994-11-07 preload.dtx v2.1e		\@begindvi: Use normal box register: why a box? ....	929
General: (DPC) Updated to use \ProvidesFile ....	564	\@begindvibox: Use normal box register: why a box? ....	910
1994-11-09 ltboxes.dtx v1.0p		\@writesetup: Modify new \AtBeginDvi concept ....	925
\@finalstrut: Revert \finalstrut to 2.09 equivalent (from ltpatch) ..	681	General: Removed old definition of \@testfp. ....	898
General: more color changes... ....	667	1994-11-14 lspace.dtx v1.2d	
1994-11-09 ltfsbas.dtx v2.1v		\\: (DPC) Macro modified ....	314
\@vpt: (DPC) macros added, from setsizes.dtx ....	432	1994-11-14 ltab.dtx v1.0i	
(DPC) reduce save stack usage latex/1742 ....	432	\tabularnewline: (DPC) Macro added ....	696

1994-11-16 fontdef.dtx v2.2h		1994-11-18 ltfsstrc.dtx v2.3m	
General: (DPC) Removed \{ and \}	545	General: \next to \reserved@f ...	455
1994-11-17 ltboxes.dtx v1.0q		\phantom: (DPC) colour support ...	637
General: \tempa to \reserved@a ...	667	(DPC) use \expandafter instead of \next ...	637
1994-11-17 ltclass.dtx v1.0l		\prime@s: (DPC) use \let@token	
General: \tempa to \reserved@a ...	791	instead of \next and \expandafter instead of \nxt ...	642
1994-11-17 ltcntrl.dtx v1.0b		\smash: (DPC) colour support ...	638
General: \tempa to \reserved@a ...	275	(DPC) use \expandafter instead of \next ...	638
1994-11-17 ltdefns.dtx v1.0g		1994-11-21 ltfloat.dtx v1.1f	
General: \tempa to \reserved@a ...	78	\endfloatbox: Added reset of minipage flag ...	765
1994-11-17 ltdirchk.dtx v1.0j		Corrected position of \outer@nobreak ...	765
General: \tempa to \reserved@a ...	1	\marginparreset: Macro added ...	769
1994-11-17 lterror.dtx v1.2h		\savemarbox: Added \setminipage	
General: \tempa to \reserved@a ...	279	etc ...	769
1994-11-17 ltfiles.dtx v1.0n		Added resetting of size and font ..	769
General: \tempa to \reserved@a ...	330	Changed to \color@vbox ...	769
1994-11-17 ltfinal.dtx v1.0o		Use \setnobreak etc ...	769
General: \tempa to \reserved@a ...	974	\setminipage: Macro added ...	763
1994-11-17 ltfloat.dtx v1.1e		\setnobreak: Macro added ...	763
General: \tempa to \reserved@a ...	756	\xfloat: Added \setminipage ...	761
1994-11-17 ltftntcmd.dtx v3.3p		Added resetting of size and font ..	761
General: \tempa to \reserved@a ...	567	Changed to \color@vbox so that large floats overflow at the bottom ...	761
1994-11-17 ltftssbas.dtx v2.1w		Missing percents reinserted after 4, 8: these are not numbers. ....	760
General: \tempa to \reserved@a ...	407	Use \setnobreak ...	761
1994-11-17 ltfsdcl.dtx v2.1m		\xympar: Changed to \color@vbox	770
General: \tempa to \reserved@a ...	485	1994-11-21 ltoutput.dtx v1.1i	
1994-11-17 ltmath.dtx v1.0o		\addtocurcol: Added \if@nobreak	
General: \tempa to \reserved@a ...	635	test before float box ...	939, 943
1994-11-17 ltmiscen.dtx v1.0v		\specialoutput: Added \if@nobreak	
General: \tempa to \reserved@a ...	614	test ...	918
1994-11-17 ltoutenc.dtx v1.6h		\topnewpage: Changed to \color@vbox ...	914
General: (DPC) \tempa to \reserved@a ...	351	1994-11-22 ltfsdcl.dtx v2.1o	
1994-11-17 ltoutput.dtx v1.1h		General: wrap long lines ...	485
General: \tempa to \reserved@a ..	898	1994-11-22 ltoutenc.dtx v1.6i	
1994-11-17 ltpictur.dtx v1.0f		General: Corrected \dots so that there's no kerning in monowidth fonts. ....	351
General: \tempa to \reserved@a ...	707	Corrected typo with \mathunderscore. ....	351
1994-11-17 ltsect.dtx v1.0i		Fixed empty accents. Again. ....	351
General: \tempa to \reserved@a ...	743	1994-11-24 ltdefns.dtx v1.2h	
1994-11-17 ltab.dtx v1.0j		\newenv: Added test for \endgraf ...	85
General: \tempa to \reserved@a ...	682		
1994-11-18 ltboxes.dtx v1.0r			
\color@vbox: macro added for color support ...	670		
1994-11-18 ltfinal.dtx v1.0n			
General: re-allow slots 127-255 ....	986		
1994-11-18 ltftssbas.dtx v2.1x			
General: (DPC) use \reserved@f not \next ...	407		
1994-11-18 ltfsdcl.dtx v2.1m			
\DeclareMathDelimiter: (DPC) \expandafter instead of \next ..	507		

1994-11-25 ltplain.dtx v1.1f General: (DPC) Comment out lots of obsolete code . . . . .	15	1994-12-01 ltfinal.dtx v1.0p General: Renamed lthyphen.* to hyphen.*. . . . .	974
1994-11-26 ltfloat.dtx v1.1b \footnote: (ASAJ) Added \protected@xdef. . . . .	775	1994-12-01 lthyphen.dtx v1.0g General: Rename lthyphen.ltx/cfg to hyphen.ltx/cfg . . . . .	972
1994-11-28 ltcntrl.dtx v1.0c General: Documentation improvements . . . . .	275	1994-12-01 ltplain.dtx v1.1g General: (DPC) More doc changes . . .	15
1994-11-30 ltfiles.dtx v1.0o \@dofilelist: Macro added . . . . .	350	1994-12-02 fontdef.dtx v2.2i General: Commented out \ldots.	
\listfiles: Use \@dofilelist . . . . .	349	ASAJ. . . . .	543
\nofiles: There is no \gobblethree. . . . .	337	1994-12-02 ltfssini.dtx v2.2c \copyright: \copyright is now in ltoutenc. ASAJ . . . . .	538
1994-11-30 ltfsbas.dtx v2.1y \fontshape: Use \@current@cmd in \@enc@update. ASAJ. . . . .	416	1994-12-02 ltlists.dtx v1.0e \@trivlist: RmS: Added check for looping . . . . .	658
1994-11-30 ltmath.dtx 1.0q General: ASAJ: \DeclareMathOperator moved to AMSLATEX. . . . .	635	1994-12-02 ltoutenc.dtx 1.7b General: Redefined \a properly. . . .	363
1994-11-30 ltmiscen.dtx v1.0w \enddocument@kernel@\warnings: (DPC) Do warnings even for \nofiles . . . . .	616	1994-12-02 ltoutenc.dtx v1.7b General: Fixed a bug with \a. . . . .	351
\enddocument: (DPC) Use \@dofilelist . . . . .	616	1994-12-04 lthyphen.dtx v1.0h General: Documentation edits for /1989 . . . . .	972
1994-11-30 ltoutenc.dtx 1.7a General: Redefined \a for the new scheme. . . . .	363	1994-12-05 ltoutenc.dtx v1.7c General: Added braces to \textcircled. . . . .	351
1994-11-30 ltoutenc.dtx v1.6g General: Removed new definitions of \patterns and \hyphenation, since encoding-specific commands now expand in the mouth. . . . .	363	1994-12-06 ltfssbas.dtx v2.1z \DeclareFontEncoding: use \nfss@catcodes . . . . .	411
1994-11-30 ltoutenc.dtx v1.7a General: Added new code for encoding-specific commands. These now expand in the mouth, which means that ligaturing and kerning can happen. . . . .	351	\nfss@catcodes: Added tab char as well . . . . .	421
Always load the enc.def file, so that the default encoding for the commands will change. . . . .	394	1994-12-08 ltoutenc.dtx v1.7d General: Added \null and \sh@ft to \b and \d. . . . .	351
Redefined \@changed@cmd to expand in the mouth. . . . .	355	1994-12-08 ltab.dtx v1.0k \array: Add \tabularnewline . . . .	695
Removed \@changed@x@mouth since \@changed@x now expands in the mouth. . . . .	355	\tabularnewline: (DPC) Made it \relax . . . . .	696
Rewrote \@text@composite so it allows an empty argument, or an argument containing lots of commands. . . . .	357	1994-12-09 ltbibl.dtx v1.1d \bibliographystyle: (DPC) Allow use in preamble. . . . .	785
		1994-12-10 ltfloat.dtx v1.1g \dblflo: Old version reinstated temporarily . . . . .	759
		\dblflo: Macro removed temporarily . . . . .	759
		Old version reinstated temporarily	759
		\setfps: Macro removed temporarily . . . . .	760
		\dblfloat: Macros reinserted temporarily . . . . .	765
		\xflo: Old version reinstated temporarily . . . . .	760

Sanitization added temporarily . . .	760	\endfilecontents: Close input check stream: latex/1487 . . . . .	824
General: Some temps reinserted temporarily . . . . .	756	1995-04-21 ltfinal.dtx v1.0q	
\fps@dbl: Macro removed temporarily . . . . .	760	General: Allow initial patch level 0 . . . . .	989
1994-12-10 ltntcmd.dtx v3.3q		1995-04-21 ltoutenc.dtx v1.7h	
\@math@egroup: Don't read arguments . . . . .	574	General: Added \null \k latex/1274 . . . . .	351
\check@nocorr@: Use \space command for comparison . . . . .	571	1995-04-22 ltfiles.dtx v1.0p	
1994-12-10 ltssdcl.dtx v2.1p		\includeonly: Allow blanks in argument . . . . .	338
\document@select@group: Surround with braces (add fourth arg) . . . . .	491	1995-04-22 ltmiscen.dtx v1.0x	
\select@group: Surround with braces (add fourth arg) . . . . .	488	General: Removed extra def of \gobble . . . . .	614
1994-12-10 ltoutenc.dtx v1.7e		1995-04-23 ltsect.dtx v1.0j	
General: Added documentation for the OML encoding. . . . .	351	\addcontentsline: Use \contentsline internally. . . . .	752
Replaced width with \width and ditto height in vrules. . . . .	351	1995-04-24 ltbibl.dtx v1.1e	
1994-12-14 ltoutenc.dtx v1.7f		\@citex: Add \mbox to undefined case: latex/1239. . . . .	784
General: Added braces to \copyright so it works unbraced in subscripts. . . . .	351	1995-04-24 ltbibl.dtx v1.1f	
Added check for math mode in \changed@cmd. . . . .	351	\bibcite: Make \onlypreamble /1388. . . . .	783
Commented out \textasciicircum, \textasciitilde, \textbackslash, \textbar, \textgreater, \texthyphenchar, \texthyphen and \textless to save memory. . . . .	351	1995-04-24 ltcntrl.dtx v1.0d	
1995-01-12 ltmath.dtx v1.2y classes		\@for: Don't expand second argument with \edef: /1317 (DPC) . . . . .	278
\eqnnum: Added \normalcolor . . . . .	647	1995-04-24 ltoutput.dtx v1.1j	
1995-03-03 ltoutenc.dtx 1.7g		\floctracemessage: Do not add to kernel unless 'trace' specified . . . . .	956
General: Corrected an error in documentation referring to the tabular rather than the tabbing environment. . . . .	363	1995-04-24 ltoutput.dtx v1.1l	
1995-04-02 ltntcmd.dtx v3.3r		\begin{vbox}: Add \vbox latex/1392 . . . . .	910
\@math@egroup: Read them again to be able to add \relax. . . . .	574	\@writeshop: Reset \\ latex/1451 (DPC) . . . . .	926
1995-04-02 ltssdcl.dtx v2.1q		1995-04-24 ltpage.dtx v1.0f	
\document@select@group: fix problem for pr/1275 . . . . .	491	\fussy: reset \emergencystretch latex/1344 . . . . .	790
\select@group: fix problem for pr/1275 . . . . .	488	1995-04-24 ltplain.dtx v1.1h	
\set@mathdelimiter: fix pr/1329 . . . . .	509	\newlanguage: Remove remaining \outer declarations. . . . .	18
1995-04-02 ltssini.dtx v2.2d		1995-04-24 ltxref.dtx v1.1e	
\not@math@alphabet: add \noexpand to second part of message . . . . .	536	\newlabel: Make \onlypreamble for /1388. . . . .	611
1995-04-21 ltclass.dtx v1.0m		1995-04-25 ltdefns.dtx v1.2i	
\DeclareOption*: Made long /1498 . . . . .	806	\check@c: Make \long for latex/1346 . . . . .	86
		\newenvironment: Parse arguments slowly but safely /1507 . . . . .	84
		1995-04-25 ltfiles.dtx v1.0q	
		\document: Removed execution of \every@size latex/1407 . . . . .	333
		1995-04-25 ltsect.dtx v1.0k	
		\dottedtocline: Added \hbox around dots. . . . .	754
		1995-04-27 ltboxes.dtx v1.0s	
		\frameb@x: Move \leavevmode for graphics/1512 . . . . .	673

\@ifframebox: Move \leavevmode for graphics/1512 . . . . .	673	1995-05-07 ltsect.dtx v1.0o General: Use \hb@xt@ . . . . .	743
\@iirsbox: Move \leavevmode for graphics/1512 . . . . .	680	1995-05-07 ltab.dtx v1.0l General: Use \hb@xt@ . . . . .	682
\@irsbox: Move \leavevmode for graphics/1512 . . . . .	680	1995-05-08 ltbibl.dtx v1.1g \@citet: Use \@firstofone . . . . .	784
\fbox: Move \leavevmode for graphics/1512 . . . . .	672	\bibitem: Removed unnecessary braces . . . . .	783
\raisebox: Move \leavevmode for graphics/1512 . . . . .	680	\nocite: Use \@firstofone . . . . .	785
1995-04-27 ltfiles.dtx v1.0r \document: Added \global to support groups in hook . . . . .	334	1995-05-08 ltdefns.dtx v1.2k \typein: Use \@firstofone . . . . .	80
1995-04-27 ltmiscen.dtx v1.0y \enddocument: \@checkend moved after hook . . . . .	615	1995-05-08 ltdefns.dtx v1.2l \typein: Remove unnecessary braces Replace \def by \let . . . . .	80
1995-04-27 ltplain.dtx v1.1i General: Move \hang and \textindent to latex209.def . . . . .	33	1995-05-08 lfsstrc.dtx v2.3n \ifnot@nil: Use \@firstofone . . . . .	468
1995-04-29 ltcntrl.dtx v1.0e General: Moved init of \protect to ltdefns.dtx . . . . .	278	1995-05-11 fontdef.dtx v2.2j General: Updates to some plain macros . . . . .	543
Removed unused defs for \@setprotect and \@resetprotect . . . . .	278	1995-05-12 ltclass.dtx v1.0n \DeclareOption*: Use \toks@ to remove need to double hash /1557	806
1995-04-29 ltdefns.dtx v1.2j \protect: Init \protect here . . . . .	90	1995-05-12 ltfloat.dtx v1.1h \footnotemark: Add \nobreak to allow hyphenation. latex/1605 . . . . .	777
1995-04-29 ltpar.dtx v1.1b General: (TO) Comments clean-up. . . . .	290	1995-05-12 ltpictur.dtx v1.0h \pictur@: Macro added for latex/1355 . . . . .	709
1995-05-02 ltsect.dtx v1.0l \dottedtocline: Don't reset to \rmfamily . . . . .	754	1995-05-12 ltvers.dtx v1.0e General: Add autoload docstrip guards Check for format older than 1 year . . . . .	39
1995-05-03 ltsect.dtx v1.0m General: TO: Promoted documentation to doc.sty standard . . . . .	743	1995-05-13 lfsstrc.dtx v2.3o General: Use single hash mark in \DeclareOption . . . . .	456
1995-05-06 ltsect.dtx 1.0n \secCntformat: Use \quad instead of \hskip . . . . .	749	1995-05-16 ltffloat.dtx v1.1i \makefnmark: Now use \textsuperscript . . . . .	774
\@sect: Added \relax after \secCntformat just in case . . . . .	747	\textsuperscript: Command added./pr1503 . . . . .	774
1995-05-07 ltboxes.dtx v1.0t General: Use \hb@xt@ . . . . .	667	\thefootnote: Streamlined parts of code. . . . .	773
1995-05-07 ltdefns.dtx v1.2k \hb@xt@: Macro added . . . . .	79	1995-05-17 ltboxes.dtx v1.0u \irsbox: Removed surplus braces . . . . .	680
1995-05-07 ltmath.dtx v1.0r General: Use \hb@xt@ . . . . .	635	1995-05-17 ltdefns.dtx v1.0o \g@addto@macro: Make long for latex/1522 . . . . .	107
1995-05-07 ltoutput.dtx v1.1m General: Use \hb@xt@ . . . . .	898	1995-05-17 ltlists.dtx v1.0g \item: Removed surplus braces . . . . .	662
1995-05-07 ltpictur.dtx v1.0g General: Use \hb@xt@ . . . . .	707	\nbitem: Removed surplus braces . . . . .	663
1995-05-07 ltplain.dtx v1.1j General: Use \hb@xt@ . . . . .	15	enumerate: Use \thr@@ and remove surplus braces . . . . .	664
		itemize: Use \thr@@ . . . . .	665
		\makefnmark: Added \normalfont.	774

\thempfootnote: Added \itshape . . . . .	773	\textsuperscript: Use \@textsuperscript . . . . .	774
1995-05-19 ltpictur.dtx v1.1a		1995-05-24 ltfssbas.dtx v3.0a	
General: Support autoloading feature	707	General: (DPC) Make file from previous file, fam.dtx 1995/05/20	
1995-05-20 ltcnts.dtx v1.1b		v2.2d . . . . .	407
\@definecounter: Streamlined code	399	\mathgroup: (DPC) No need to redefine \newfam as not outer . . .	407
\fnsymbol: Allowing both text and math . . . . .	403	1995-05-24 ltfsscmp.dtx v3.0a	
\fnsymbol: Streamlined code . . . . .	402	General: (DPC) Make file from previous file, fam.dtx 1995/05/20	
1995-05-20 ltcnts.dtx v1.1c		v2.2d . . . . .	480
\@definecounter: And do it right . . . . .	399	1995-05-24 ltfssdcl.dtx v3.0a	
1995-05-20 ltfloat.dtx v1.1k		General: (DPC) Make file from previous file, latint.dtx 1995/05/21	
\@makefnmark: Moved \normalfont back and use \@textsuperscript . . . . .	774	v2.1t . . . . .	485
Moved \normalfont to \textsuperscript . . . . .	774	1995-05-24 ltfssini.dtx v3.0a	
\textsuperscript: Use \normalfont. . . . .	774	General: (DPC) Make file from previous file, lfonts.dtx 1995/05/23	
1995-05-21 ltfssdcl.dtx v2.1t		v2.2e . . . . .	515
\DeclareMathRadical: Allow for undefined cs names . . . . .	510	\cal: (DPC) Remove definition . . . . .	541
1995-05-21 ltlists.dtx v1.0f		\mit: (DPC) Remove definition . . . . .	541
General: Moved to doc.sty standard	650	1995-05-24 ltfssrc.dtx v3.0a	
1995-05-21 ltmath.dtx v1.0r		General: (DPC) Make file from previous file, tracefnt 1995/05/16	
\sqrt: Use \sqrt sign . . . . .	644	v2.3o . . . . .	455
General: Remove \mathhexbox from this file . . . . .	640	1995-05-24 ltfssrc.dtx v3.0b	
Update some plain macros . . . . .	635	General: (DPC) Fix \ProvidesFile usage . . . . .	455
\lefteqn: Use \rlap . . . . .	646	1995-05-25 ltclass.dtx v1.0p	
\rat: Use \sqrt sign instead of \sqrt . . . . .	637	\endfilecontents: Delete \filecontents after preamble . . . . .	824
1995-05-21 ltoutenc.dtx v1.7h		1995-05-25 ltfilehook.dtx v1.0t	
\inmathwarn: Added several \onlypreamble . . . . .	355	\unquote@efilef@und: (CAR) added \long . . . . .	858
1995-05-21 ltoutenc.dtx v1.7j		1995-05-25 ltfiles.dtx v1.0s	
General: Updated some plain macros	367	\document: Added check for \topskip zero . . . . .	334
1995-05-21 lplain.dtx v1.1j		1995-05-25 ltfiles.dtx v1.0t	
General: Moved some code to other files . . . . .	15	\iffileonpath: (CAR) added \long	345
1995-05-22 lplain.dtx v1.1k		\document: Corrected typo . . . . .	334
General: Definitions of \footins and \footnoterule moved to ltfloat. . . . .	34	\IfFileExists@: (CAR) added \long	344
1995-05-22 ltab.dtx v1.1a		\nofiles: (CAR) added \long . . . . .	337
General: Support autoloading feature	682	\protected@write: (CAR) added \long . . . . .	337
1995-05-23 ltfssini.dtx v2.2e		1995-05-25 ltffloat.dtx v1.1m	
\newfont: Font assignment made local again. . . . .	537	\savebox@: (CAR) Reset settings moved to hook . . . . .	769
1995-05-24 ltdefns.dtx v1.1l		\@xfloat: (CAR) Reset settings moved to hook . . . . .	761
\newif: (DPC) New implementation . . . . .	85	1995-05-25 ltlists.dtx v1.0i	
1995-05-24 ltdefns.dtx v1.2m		\endtrivlist: Macros moved from ltspace.dtx . . . . .	659
\typein: (DPC) New implementation . . . . .	80		
1995-05-24 ltfloat.dtx v1.1l			
\@textsuperscript: Command added. . . . .	774		
General: Moved definition of \footins and \footnoterule from lplain. . . . .	773		

1995-05-25	ltmath.dtx	v1.3c classes		
\eqnnum: replace				
\reset@font\rmfamily with				
\normalfont (PR 1578) . . . . .				647
1995-05-25	ltspace.dtx	v1.2f		
\vbsphack: (CAR) not used so				
'removed' . . . . .				320
\vspacer: (CAR) \restorepar				
added to avoid possible infinite tail				
recursion caused by a typo in the				
argument . . . . .				323
(CAR) macros modified to be more				
efficient . . . . .				323
General: Macros moved to ltlists.dtx				310
1995-05-26	ltdefns.dtx	v1.2n		
\gobblefour: (CAR) Added \longs				87
1995-05-26	ltmath.dtx	v1.0s		
\eqnnum: Removed \rmfamily (PR				
1578), replaced \reset@font with				
\normalfont . . . . .				644
1995-05-26	ltpage.dtx	v1.0g		
\ps@plain: removed \rmfamily (PR				
1578) . . . . .				788
1995-05-27	ltfssbas.dtx	v3.0b		
\mathgroup: (FMi) But a need to				
define \new@mathgroup . . . . .				407
1995-06-05	fontdef.dtx	v2.2k		
General: Moved math commands from				
ltoutenc.dtx. . . . .				561
1995-06-05	ltfinal.dtx	v1.0r		
General: Added \MakeUppercase and				
\MakeLowercase. . . . .				974
1995-06-05	ltoutenc.dtx	v1.7k		
\inmathwarn: Removed				
\protected@cmd and replaced with				
explicit \noexpand. . . . .				355
General: Allowed				
\ProvideTextCommandDefault				
after the preamble. . . . .				357
Commented out \textless and				
\textgreater. . . . .				365
Moved math commands to				
fontdef.dtx. . . . .				367
Save some tokens in				
\textvisible{space} and				
\textunderline{score}. . . . .				365
1995-06-06	ltfinal.dtx	v1.0s		
General: Made \MakeUppercase and				
\MakeLowercase brace their				
argument. . . . .				974
1995-06-09	ltoutenc.dtx	v1.7l		
\DeclareTextComposite: Rewrote				
\DeclareTextComposite to define				
the composite as a no-argument				
command rather than a				
two-argument command. . . . .				358
1995-06-11	ltspace.dtx	v1.2g		
\restorecr: (CAR) \relax added to				
stop silent eating of *. . . . .				328
1995-06-13	ltfinal.dtx	v1.0t		
General: Add patch level string more				
carefully . . . . .				989
Call \errorstopmode . . . . .				991
1995-06-13	ltpictur.dtx	v1.1b		
General: Use \ProvidesFile in				
autoload . . . . .				707
1995-06-14	lttab.dtx	v1.1b		
General: Use \ProvidesFile in				
autoload . . . . .				682
1995-06-15	ltfssbas.dtx	v3.0c		
General: (DPC) minor documentation				
changes . . . . .				407
1995-06-15	ltfsscmp.dtx	v3.0b		
General: (DPC) minor documentation				
edits . . . . .				480
1995-06-15	ltfssdcl.dtx	v3.0b		
General: (DPC) minor documentation				
changes . . . . .				485
1995-06-19	ltbibl.dtx	v1.1h		
\bincite: Call \newlabel so				
repeated keys produce better				
warning. . . . .				783
1995-06-19	ltclass.dtx	v1.0q		
\documentclass: Don't redefine				
\usepackage in compat mode for				
/1634 . . . . .				811
1995-06-19	ltxref.dtx	v1.1e		
\newlabel: Use \newlabel to share				
code with \bincite . . . . .				611
1995-06-28	ltfssini.dtx	v3.0b		
General: (DPC) Fix documentation				
typos . . . . .				515
1995-06-28	ltmath.dtx	v1.0t		
General: minor doc edits . . . . .				635
1995-07-02	ltplain.dtx	v1.1n		
General: Removed surplus 'by' and '='				
in various places . . . . .				15
\offinterlineskip: Replaced 1000 by				
\@m . . . . .				31
\showoutput: Use \showoverfull to				
save space . . . . .				34
\tracingall: Use \showoutput to				
save space . . . . .				35
1995-07-03	ltdefns.dtx	v1.2o		
\set@typeset@protect: Use				
\@typeset@protect for init . . . . .				90

1995-07-03 ltfntcmd.dtx v3.3s		1995-07-14 ltbibl.dtx v1.1i
\@st@ic: Use clean interface for jump ..... 573		\bibcite: Remove \onlypreamble so still defined in new \enddocument 783
1995-07-05 ltfntcmd.dtx v3.3s		1995-07-14 ltxref.dtx v1.1g
\@st@ic: Renamed from \test@next 573		\newlabel: Remove \onlypreamble so still defined in new \enddocument ..... 611
1995-07-05 ltspace.dtx v1.2h		1995-07-19 ltfssini.dtx v3.0d
\@newline: Use \break ..... 316		General: (DPC) TeX2 support .... 541
\@no@pgbk: Macro replaces \@pgbk and \@nopgbk ..... 314		1995-07-20 ltboxes.dtx v1.0v
\@nopagebreak: Reimplemented both using \@no@pgbk ..... 313		\@isavebox: Use \sbox ..... 671
1995-07-09 ltcntrl.dtx v1.0f		\@isavepicbox: Use \sbox ..... 671
\@iforloop: Reimplemented using Kabelschacht method ..... 278		1995-07-21 ltoutput.dtx v1.1o
\@iwhiledim: Reimplemented using Kabelschacht method ..... 276		\@writesetup: Command added ... 925
\@iwhilenum: Reimplemented using Kabelschacht method ..... 276		New, experimental, versions: need in-lining ..... 925
\@iwhilesw: Reimplemented using Kabelschacht method ..... 276		1995-08-09 ltmath.dtx v1.0u
\@tfor: Reimplemented using Kabelschacht method ..... 278		General: Added code for class options leqno and fleqn ..... 647
1995-07-09 ltlists.dtx v1.0j		1995-08-11 ltlength.dtx v1.1b
\@enumerate: Use \expandafter .... 664		General: Doc typos fixed for latex/753 ..... 405
\@itemize: Use \expandafter .... 665		1995-08-16 ltcntrl.dtx v1.0g
1995-07-12 ltpictur.dtx v1.1d		\@break@tfor: Made long ..... 278
General: allow 2e commands in 209 mode. latex/1737 ..... 707		\@forloop: Made defs long ..... 278
1995-07-13 ltdefns.dtx v1.0p		\@fornoop: Made defs long ..... 278
General: Updates to documentation .. 78		\@iforloop: Made defs long ..... 278
1995-07-13 ltfiles.dtx v1.0u		\@iwhiledim: Made defs long ..... 276
General: Updates to docu ..... 330		Removed \@whilenoop ..... 276
1995-07-13 ltfssbas.dtx v3.0d		\@iwhilenum: Made defs long ..... 276
\@defaultsubs: macro added .... 427		Removed \@whilenoop ..... 276
\@defaultsubs: macro added .... 427		\@iwhilesw: Removed \@whileswnoop ..... 276
General: minor documentation changes ..... 407		\@tfor: Made defs long ..... 278
\@wrong@fontshape: Change a macro not a switch to flag default font substitutions ..... 426		1995-08-16 ltfiles.dtx v1.0v
1995-07-13 ltmiscen.dtx v1.0z		\document: set \@maxdepth ..... 334
\@centercr: Use \nobreak ..... 626		set \do globally ..... 334
\@enddocument@kernel@warnings: Use \@defaultsubs instead of switch 617		set \topskip globally ..... 334
\@writefile: Added missing percent and use \relax in the THEN case 620		1995-08-24 ltfssbas.dtx v3.0f
\@xobeysp: Use \nobreak ..... 629		General: Added autoload code .... 407
General: Improve Documentation .. 614		1995-08-24 lfsstrc.dtx v3.0c
\@enddocument: Set \@setckpt to \@gobbletwo instead of defining it by hand ..... 615		General: Macro \gobble@font@spec removed ..... 469
Shorten redefinition of \bibcite and \newlabel ..... 616		\tryis@simple: ..... 476

1995-09-14 ltplain.dtx v1.1o		1995-10-11 ltoutput.dtx v1.1r
General: Moved <code>\multispan</code> to ltab.dtx . . . . .	15	<code>\clearpage</code> : Added a check so that it does not lose the argument of <code>\twocolumn[...]</code> . . . . .
1995-09-14 ltab.dtx v1.1c		911
<code>\cline</code> : (DPC) New implementation	705	1995-10-16 ltbibl.dtx v1.1j
1995-09-15 ltfssini.dtx v3.0e		<code>\cite</code> : (DPC) Make robust . . . . .
General: (DPC) Modify TeX2 message . . . . .	541	783
1995-09-19 ltmiscen.dtx v1.1a		1995-10-16 ltboxes.dtx v1.0w
<code>\verb</code> : Put <code>\@noligs</code> after <code>\verbatim@font</code> where it belongs.	634	General: Clarify makebox description
1995-10-01 ltfiles.dtx LaTeXe		667
<code>\@addtofilelist</code> : Macro added . . .	349	1995-10-16 ltdefns.dtx v1.2u
1995-10-02 ltdefns.dtx v1.2q		<code>\@ifstar</code> : (DPC) New implementation, for /1910 . . . . .
<code>\@ifnch</code> : Use <code>\@let@token</code> for internal/924, save <code>\reserved@e</code> .	103	103
<code>\@ifnextchar</code> : Use <code>\@let@token</code> . . .	102	<code>\new@command</code> : (DPC) Use <code>\@testopt</code> /1911 . . . . .
<code>\@protected@testopt</code> : Macro added .	83	82
<code>\@testopt</code> : Macro added . . . . .	82	<code>\new@environment</code> : (DPC) Use <code>\@testopt /1911</code> . . . . .
<code>\@xargdef</code> : New implementation, using <code>\@test@opt</code> . . . . .	82	84
1995-10-03 fontdef.dtx v2.2l		<code>\typein</code> : (DPC) Use <code>\@testopt /1911</code>
General: <code>\@@sqrt</code> from patch file for /1701 . . . . .	543	80
1995-10-03 ltdefns.dtx v1.2r		1995-10-16 ltfssini.dtx v3.0f
<code>\typein</code> : Add missing <code>\@typein</code> for /1710 (from patch file) . . . . .	80	<code>\reset@font</code> : Added <code>\relax</code> after <code>\usefont</code> , as the latter eats up spaces. . . . .
1995-10-03 ltpictur.dtx v1.1e		538
General: New autoload code . . . . .	707	1995-10-16 ltmath.dtx v1.0y
1995-10-04 ltfssbas.dtx v3.0g		<code>\@yeqnqr</code> : (DPC) Use <code>\@testopt</code> /1911 . . . . .
General: Modify autoload code . . . . .	407	645
1995-10-04 lftsstrc.dtx v3.0d		<code>\sqrt</code> : (DPC) Make robust /1808 . .
General: (DPC) Modify autoload code . . . . .	455	644
1995-10-04 ltab.dtx v1.1d		1995-10-16 ltspace.dtx v1.2j
General: Modify autoload support .	682	<code>\nolinebreak</code> : (DPC) Use <code>\@testopt</code> /1911 . . . . .
1995-10-06 ltfiles.dtx v1.0w		313
<code>\@missingfileerror</code> : Autoload error	347	<code>\nopagebreak</code> : (DPC) Use <code>\@testopt</code> /1911 . . . . .
1995-10-09 lterror.dtx v1.2j		313
General: Modify autoload support .	279	1995-10-16 lthm.dtx v1.0g
1995-10-09 ltoutenc.dtx v1.7m		General: Revert to previous <code>\newtheorem</code> behaviour . . . . .
<code>\@inmathwarn</code> : Autoload error . . .	356	739
1995-10-10 ltfssbas.dtx v3.0h		1995-10-17 ltclass.dtx v1.0r
<code>\showhyphens</code> : Use <code>\normalfont</code> and make colour safe, and autoloadable . . . . .	430	<code>\@providesfile</code> : Delay definition of <code>\ProvidesFile</code> till lfinal . . . . .
1995-10-10 ltfsdcl.dtx v3.0c		805
<code>\non@alpherr</code> : (DPC) autoload error message . . . . .	489	<code>\ProcessOptions*</code> : Reset <code>\CurrentOption</code> for graphics/1873 . . . . .
1995-10-10 ltplain.dtx v1.1r		809
General: Autoload tracing code . . . .	15	1995-10-17 ltdirchk.dtx v1.0l
1995-10-10 lthm.dtx v1.0f		General: Modify initex version of <code>\ProvidesFile</code> . . . . .
General: Make <code>\newtheorem</code> ‘only preamble’ . . . . .	739	4
		1995-10-17 ltfinal.dtx v1.0v
		<code>\@providesfile</code> : reset macro . . . .
		990
		<code>\reserved@b</code> : reset here after the <code>\input</code> above . . . . .
		989
		1995-10-17 ltplain.dtx v1.1s
		<code>\reject</code> : Move <code>\supereject</code> to compat file . . . . .
		32
		1995-10-17 ltab.dtx v1.1e
		<code>\@cline</code> : (DPC) Use <code>\@multicnt</code> . .
		705
		<code>\@multispan</code> : (DPC) Macro added.
		705
		1995-10-19 ltfinal.dtx v1.0w
		<code>\@filelist</code> : Move after <code>\reserved@a</code> setting:-) . . . . .
		990

1995-10-20	ltbibl.dtx v1.1k		\@setref: Switch for refundefined renamed . . . . .	611
	\@citex: Removed refundefined flag	784		
	\nocite: Removed refundefined flag	785	\if@multiplelabels: Macro removed	611
1995-10-20	ltclass.dtx v1.0s			
	\@begindocumenthook: Make setting conditional, for autoload version	822	1995-10-25 ltalloc.dtx v1.1b General: General doc improvements	273
1995-10-20	ltfssbas.dtx v3.0i			
	General: (DPC) Modify autoload code, change \undefined . . . . .	407	1995-10-25 ltfloat.dtx v1.1n \endfloatbox: (CAR) macro added: to unify code for double and single versions . . . . .	765
1995-10-20	ltfsstrc.dtx v3.0e		\enddblfloat: (CAR) unify code for double and single versions . . . . .	764
	General: (DPC) Modify autoload code . . . . .	455	\endfloat: (CAR) unify code for double and single versions . . . . .	763
1995-10-22	ltfssbas.dtx v3.0j			
	General: (RmS) New size function macro \genb@sfcnt needs to be disabled at \document. . . . .	407	1995-10-25 ltidxglo.dtx v1.1d General: Doc cleanup . . . . .	779
1995-10-22	ltfsstrc.dtx v3.0f			
	General: Added 'genb' and 'sgenb' size functions to support new DC font naming scheme. . . . .	455	1995-10-25 ltsect.dtx v1.0q \subparagraphmark: Use \let not \def to save space. . . . .	751
1995-10-23	lttab.dtx v1.1f			
	\@settab: (CAR) Ensure that \hightab increases by at most one . . . . .	689	1995-10-27 ltpictur.dtx v1.1f General: Move initialization to kernel from autoload file . . . . .	734
	\@startline: (CAR) Ensure that \nxttabmar is never larger than \hightab . . . . .	687	1995-10-31 ltboxes.dtx v1.0x \finalstrut: Add \nobreak in horiz mode to allow hyphenation. internal/1931 . . . . .	681
	\poptabs: (CAR) Ensure that \curtab is never larger than \hightab . . . . .	690		
	\tabbing: (CAR) Make \hightab consistently a local variable . . .	689	1995-11-01 fontdef.dtx v2.2m General: add \nfss@catcodes for internal/1932 . . . . .	546
1995-10-24	ltfiles.dtx v1.1a			
	\document: Removed multiplelabels switch . . . . .	333	1995-11-01 ltdirchk.dtx v1.0n General: Initialise \@addtolist to \gobble . . . . .	4
	Removed refundefined switch . . .	333		
1995-10-24	ltfssbas.dtx v3.0k		1995-11-01 ltfinal.dtx v1.0x General: (DPC) Switch meaning of \addtolist for cfg files . . .	980
	\@defaultsubs: macro removed . .	427		
	\wrong@fontshape: Make this code inline since it happens only here	426	1995-11-01 ltfssbas.dtx v3.0m \DeclareFontShape@: (DPC) Test for \relax not \undefined, internal/1933 . . . . .	408
1995-10-24	ltmisen.dtx v1.1b			
	\@enddocument@kernel@warnings: Changed logic for producing warning messages and removed switch . . . . .	617	1995-11-01 ltfssini.dtx v3.0g General: (DPC) Switch meaning of \addtolist for cfg files . . .	541
	Use \@refundefined instead of switch . . . . .	617		
1995-10-24	ltxref.dtx v1.1h		1995-11-02 ltfssbas.dtx v3.0n \wrong@fontshape: (DPC) Remove extra space with \string for latex/1676 . . . . .	425
	\@multiplelabels: Switch for multiplelabels removed . . . . .	611		
	\newl@bel: Switch for multiplelabels replaced by inline code . . . . .	611	1995-11-02 ltoutenc.dtx v1.7n General: Changed internal name \a to \tabacckludge to protect against redefinition by malicious users. . .	363
	\@refundefined: Switch for refundefined replaced . . . . .	610		
			\@doendpe: Enclosed \setbox0 assignment by a group so that it leaves the contents of box 0 intact. . . . .	660

1995-11-07 ltoutenc.dtx v1.7o		1995-11-29 ltoutenc.dtx v1.7t
General: Added <code>\leavevmode</code> at start of <code>\c</code> , otherwise the output routine might be invoked within the macro. . . . .	367	General: Added <code>\textasciicircum</code> , <code>\textasciitilde</code> , <code>\textbackslash</code> , <code>\textbar</code> , <code>\textgreater</code> and <code>\textless</code> . . . . .
Changed <code>\char32</code> to <code>\@xxxii</code> (two tokens less). . . . .	368	Added <code>\textasciicircum</code> , <code>\textasciitilde</code> , <code>\textregistered</code> and <code>\texttrademark</code> . . . . .
Replaced octal number 27 by decimal number 23 to protect against the quote character being active. . . . .	368	Added <code>\textbackslash</code> and <code>\textbar</code> . . . . .
Replaced some 0's by <code>\z@</code> (faster).	368	Added <code>\textless</code> and <code>\textgreater</code> . . . . .
1995-11-10 ltoutput.dtx v1.1s		1995-12-01 ltoutenc.dtx v1.7u
<code>\@shipoutsetup</code> : Command removed	925	General: Made <code>\\$S</code> a Default, rather than having the default point to the OT1 definition. . . . .
<code>\@writesetup</code> : Command removed .	925	1995-12-04 ltspace.dtx v1.2k
In-lined . . . . .	925	<code>\nobreakspace</code> : (Macro added . . . . .
1995-11-14 ltclass.dtx v1.0t		1995-12-04 ltspace.dtx v1.2l
<code>\@unprocessedoptions</code> : Allow empty option . . . . .	823	<code>\@obeysp</code> : (braces added to definition of tilde . . . . .
<code>\@loadwithoptions</code> : macro added .	812	1995-12-04 preload.dtx v2.4e
<code>\LoadClassWithOptions</code> : macro added . . . . .	812	General: Ulrik Vieth. added 12pt OMS and OML preloads /1989 .
<code>\RequirePackageWithOptions</code> : macro added . . . . .	812	1995-12-05 ltdefns.dtx 1.2w
1995-11-17 lfssbas.dtx v3.0m		<code>\@unexpandable@protect</code> : Removed <code>\unexpandable@noexpand</code> as never used. internal/1733 . . . . .
<code>\@wrong@font@char</code> : (DPC) Macro added. latex/1676 . . . . .	427	1995-12-05 ltfiles.dtx v1.1c
<code>\define@newfont</code> : Redefine <code>\typeout</code> latex/1676 . . . . .	420	<code>\document</code> : <code>\ignorespaces</code> added for latex/1933 . . . . .
<code>\wrong@fontshape</code> : Support <code>\@wrong@font@char</code> latex/1676 .	425	1995-12-05 ltfloat.dtx v1.1n
1995-11-17 ltoutenc.dtx v1.7p		<code>\@textsuperscript</code> : Use <code>\ensuremath</code> for latex/1984. . . . .
<code>\UseTextSymbol</code> : Support <code>\@wrong@font@char</code> latex/1676 .	360	1995-12-05 ltoutenc.dtx v1.7v
1995-11-18 ltoutenc.dtx v1.7q		<code>\@inmathwarn</code> : Changed <code>\TextSymbolUnavailable</code> text .
<code>\UseTextSymbol</code> : Modify message slightly . . . . .	360	1995-12-06 lfssbas.dtx v3.00
1995-11-21 fontdef.dtx v2.2n		<code>\nfss@catcodes</code> : Reset hash, for typeouts etc in fd files . . . . .
General: Incorporate changed figures, as in plain.tex . . . . .	559	1995-12-07 ltbibl.dtx v1.1l
1995-11-27 lfssbas.dtx v3.0n		<code>\@citex</code> : Restored name of <code>\G@refundefinedtrue</code> . . . . .
<code>\nfss@catcodes</code> : Reset hash, for definitions in fd files . . . . .	422	1995-12-07 ltfloat.dtx v1.1m
1995-11-28 ltfloat.dtx v1.1n		<code>\@textsuperscript</code> : Move <code>\m@th</code> out of the <code>\ensuremath</code> for latex/1984. . . . .
General: documentation fixes . . . . .	756	1995-12-07 ltxref.dtx v1.1i
1995-11-28 lfsstrc.dtx v3.0g		<code>\@setref</code> : Switch for refundefined restored . . . . .
General: documentation fixes . . . . .	455	<code>\G@refundefinedtrue</code> : Renamed (back) from <code>\G@refundefined</code> . . . . .
1995-11-28 ltoutenc.dtx v1.7r		
General: Added math mode checks to text commands. . . . .	355	
doc fixes . . . . .	351	
Renamed <code>\@changed@x@err</code> to <code>\TextSymbolUnavailable</code> . . . . .	355	

1995-12-11 ltoutenc.dtx v1.7w		1996-05-21 ltoutenc.dtx v1.7y	
General: Modified \copyright . . . . .	365	General: Corrected error message (CAR) . . . . .	394
1995-12-13 ltdefns.dtx 1.2x		1996-05-21 ltsect.dtx v1.0s	
\#: Documentation changed. . . . .	105	\@sect: (DPC) Added extra braces for internal/2148 . . . . .	747
1996-01-10 ltfiles.dtx v1.1d		(DPC) Moved brace to allow commands like \MakeUppercase in 6th argument. Changed \par to \endgraf to allow non-long commands. internal/2148 . . . . .	747
\@iffileonpath: Change argument handling to not require doubled hash. latex/2024 . . . . .	345	\@ssect: (DPC) Added extra braces for internal/2148 . . . . .	750
1996-01-20 ltidxglo.dtx v1.1e		(DPC) Moved brace to allow commands like \MakeUppercase in 4th argument. Changed \par to \endgraf to allow non-long commands. internal/2148 . . . . .	750
\makeglossary: Make no-op after use pr/2048 . . . . .	780	1996-05-23 ltoutenc.dtx v1.7z	
\makeindex: Make no-op after use pr/2048 . . . . .	780	\@strip@args: \expandafter added to match other changes for latex/2133 . . . . .	360
1996-01-20 ltspace.dtx v1.2m		\add@accent: macro added. latex/2133 . . . . .	357
\vspace: Made robust . . . . .	323	\DeclareTextAccent: Reimplemented using \add@accent to save space latex/2133 . . . . .	357
1996-03-25 ltmath.dtx v1.1a		\DeclareTextCompositeCommand: Modified to cope with new \add@accent command: required removal of check for one argument-command . . . . .	358
\@ensuredmath: Macro added for amslatex/2104 . . . . .	647	1996-05-24 ltoutput.dtx v1.1t	
\ensuremath: Reimplement for amslatex/2104 . . . . .	646	\@specialoutput: Check that \colroom is less than \vsize, indicating that a float has been added . . . . .	916
1996-04-18 ltpage.dtx v1.0i		Cut-off point changed to 1.5\baselineskip . . . . .	916
General: Improve documentation . . . . .	787	\@topnewpage: Cut-off point changed to 2.5\baselineskip . . . . .	915
1996-04-22 ltmiscen.dtx v1.1c		1996-05-25 ltoutput.dtx v1.1u	
General: Improve Documentation . . . . .	614	\@specialoutput: Correct the above check . . . . .	916
1996-04-22 ltspace.dtx v1.2n		1996-06-03 ltmiscen.dtx v1.1d	
General: Documentation Improvements . . . . .	310	\overline: Exchanged the following two code lines so that \dospecials cannot reset the category code of characters handled by \noligs. . . . .	630
1996-04-22 lttab.dtx v1.1g		General: Move setting of verbatim font and \noligs. . . . .	614
\@tabclassz: (DPC) Extra \hskip keeps tabcolsep in empty columns internal/2122 . . . . .	702	\verb: Put setting of verbatim font after \dospecials so that \dospecials cannot reset the	
1996-04-23 ltcnts.dtx v1.1d			
General: Documentation improvements . . . . .	397		
1996-04-24 ltfiles.dtx v1.1e			
\document: (DPC) Reset \AtBeginDocument eg for latex/1297 . . . . .	333		
1996-05-08 lfsstrc.dtx v3.0h			
\math@egroup: Use \bgroup instead of \begingroup to match a kernel change made in 1994!! . . . . .	467		
1996-05-09 lftntcmd.dtx v3.3t			
\check@icr: Default definitions added . . . . .	571		
1996-05-17 fontdef.dtx v2.2o			
General: \@sqrt removed, at last . . . . .	543, 559		
1996-05-17 ltfiles.dtx v1.1f			
\nofiles: added \write to \protected@write for latex/2146	337		
1996-05-18 ltoutenc.dtx v1.7x			
General: Produce error if encoding not found. pr/2054 . . . . .	394		

category code of characters handled by \noligs. ....	634	\nfss@catcodes: omit \relax as not needed ....	421
1996-06-10 ltboxes.dtx v1.0y \parboxto: (DPC) Changed \endgraf to \par ..... 675		1996-07-26 ltfssdcl.dtx v3.0e \init@restore@version: Removed \ifrestore@version switch and replaced by \init@restore@version ..... 489	
1996-06-10 ltsect.dtx v1.0t \sect: (DPC) Changed \endgraf to \par ..... 747		1996-07-26 lfsstrc.dtx v3.0i \init@restore@glb@settings: macro added replacing \ifinmath switch ..... 466	
1996-06-10 ltsect.dtx v1.0t \sect: (DPC) Changed \endgraf to \par ..... 750		1996-07-26 ltlsts.dtx v1.0l \item: Remove unnecessary \global before \minipage... ..... 661	
1996-06-13 ltdirchk.dtx v1.0r General: documentation improvements mainly from internal/2174 ..... 1		Remove unnecessary \global before \nobreak... ..... 662	
1996-06-14 lttab.dtx v1.1h \tabclassz: (DPC) Change both\zskip to 1sp for latex/2160 ..... 702		1996-07-26 ltmath.dtx v1.1b General: Removed \global before \ignoretrue in various places. . 635	
1996-06-22 ltspace.dtx v1.2o General: Documentation of problems added ..... 310		1996-07-26 ltmiscen.dtx v1.1e \ignorefalse: put \global into definition ..... 615	
1996-07-10 ltfinal.dtx v1.0y \toks: Free up memory from scratch registers /2213 ..... 989		\begin: remove \global before \ignore... ..... 621	
1996-07-19 ltoutenc.dtx v1.8a \stripargs: Use char 0 not @ as carrier for \lowercase /2197 . . 360		\end: remove \global before \ignore... ..... 624	
1996-07-26 ltboxes.dtx v1.0z \ifminipage: put \global into definition ..... 676		\ignorespacesafterend: user level macro added ..... 615	
1996-07-26 ltclass.dtx v1.0u \classoptionslist: made only preamble ..... 796		1996-07-26 ltoutput.dtx v1.1v \testfp: remove \global before \test... ..... 960	
\unusedoptionlist: made only preamble ..... 796		\tryfc: remove \global before \test... ..... 933	
1996-07-26 ltdefns.dtx v1.2y \reargdef: third arg picked up by \argdef ..... 83		\ztryfc: remove \global before \test... ..... 934	
\renewcommand: use \noexpand instead of \string ..... 84		General: put \global into definition remove \global before \test... . 908	
use \relax in place of empty arg . 84		\clearpage: add number of missing percents ..... 911	
\renewenvironment: use \relax in place of empty arg ..... 85		1996-07-26 lplain.dtx v1.1t \sh@ft: replace \dimen\z@ by \dimen@ 33	
1996-07-26 ltfloat.dtx v1.1n \endfloatbox: remove unnecessary \global before \minipage... . 765		1996-07-26 ltsect.dtx v1.0u \starttoc: removed \global before \nobreak... ..... 752	
\savemarbox: remove unnecessary \global before \minipage... . 769		\csect: Removed \global before \nobreak... ..... 748	
\setminipage: remove unnecessary \global before \minipage... . 763		1996-07-26 ltspace.dtx v1.2p \ifnobreak: put \global inside definition ..... 316	
\setnobreak: remove unnecessary \global before \nobreak... . 763		1996-07-27 lfssbas.dtx v3.0q General: \ifinmath switch removed 419	
1996-07-26 ltssbas.dtx v3.0p \DeclareMathSizes: use faster \if test ..... 414		1996-07-27 ltspace.dtx v1.2q General: Further documentation of problems ..... 310	

1996-07-27 ltspace.dtx v1.2r		1996-10-21 ltab.dtx v1.1i	
General: Correct documentation of problems . . . . .	310	\array: Use \set@typeset@protect . . . . .	695
1996-08-02 ltfloat.dtx v1.1o		General: Moved the code associated with \cmkpream into the group provided by the box, for robustness (latex/2183) . . . . .	694
\@xmpar: Remove \global before \@ignore . . . . .	770	\multicolumn: Make \multicolumn long (latex/2180) . . . . .	697
1996-08-02 ltsect.dtx v1.0v		\tabbing: Moved the \indent so that the \everypar can remove it when necessary; this is needed because the code for items in lists has changed (see pr/22111) . . . . .	689
\@afterheading: Removed \global before \nobreak . . . . .	750	1996-10-23 llists.dtx v1.0m	
1996-08-02 ltspace.dtx v1.2s		\item: \nobreak... moved into the \everypar and not executed unconditionally, see above . . . . .	662
\@EspHack: Remove \global before \@ignore . . . . .	319	\kern... changed to \setbox... . . . . .	662
1996-08-25 lfssbas.dtx v3.0r		Added setting of \clubpenalty and set \nobreakfalse only when necessary . . . . .	662
\nfss@catcodes: Reset the acute, grave and double quote chars as well . . . . .	422	1996-10-23 ltdirchk.dtx v1.0t	
1996-09-21 ltoutput.dtx v1.1w		\@xsect: Replaced \hskip... with \setbox... as used in \@afterheading . . . . .	748
\@writesetup: Added \@parboxrestore and made consequent deletions: wait for the howls of protest . . . . .	925	1996-10-24 ltboxes.dtx v1.1a	
1996-09-25 ltdirchk.dtx v1.0t		\array@parboxrestore: Added local settings of flags: dangerous! . . . . .	675
General: Move ltxcheck to separate file	14	\@iiiminipage: Use it or lose it (@setminpage): Frank will want to lose it . . . . .	677
1996-09-28 ltmiscen.dtx v1.1f		1996-10-24 ltfloat.dtx v1.1p	
\@xobeysp: Moved to ltspace.dtx . . .	629	\@floatboxreset: Added local settings of flags: dangerous! . . . . .	763
1996-09-28 ltspace.dtx v1.2t		\@marginparreset: Added local settings of flags: dangerous! . . . . .	769
\@xobeysp: Moved from ltmiscen.dtx and redefined to use \nobreakspace . . . . .	326	\@xfloat: Added \nодокумент to trap floats in the preamble . . . . .	760
1996-09-29 ltfiles.dtx v1.1g		1996-10-24 ltoutput.dtx v1.1z	
\document: Added disabling of \nодокумент . . . . .	334	\@addtocurcol: Added \nobreak, etc as appropriate . . . . .	939, 943
1996-09-29 ltoutput.dtx v1.1x		\@specialoutput: Added \nobreak as appropriate . . . . .	918
\newpage: Checks for noskipsec and inlabel added . . . . .	912	\@topnewpage: Added \nодокумент to trap \twocolumn in the preamble . . . . .	914
1996-09-29 ltsect.dtx 1.0w		\newpage: Better checks for noskipsec and inlabel added, plus nobreak . . . . .	912
\@noskipsectrue: Added documentation . . . . .	744	1996-10-25 llists.dtx v1.0n	
1996-09-30 ltoutput.dtx v1.1y		\endtrivlist: Change \indent to \leavevmode . . . . .	659
\newpage: Checks for noskipsec and inlabel removed pending further tests . . . . .	912	Reset flags explicitly . . . . .	659
1996-10-04 ltclass.dtx v1.0v		1996-10-25 ltoutput.dtx v1.2a	
\RequirePackageWithOptions: Reset \@unprocessedoptions for /2269	812	\newpage: Reset all flags explicitly . . . . .	912
1996-10-05 ltfiles.dtx v1.1h			
\clubpenalty: Added setting its value . . . . .	332		
1996-10-08 lfntcmd.dtx v3.3u			
\DeclareTextFontCommand: Removed \check@icr when in vmode since it causes various errors (see pr/2157) . . . . .	569		

1996-10-26 ltlists.dtx v1.0o		1996-11-18 ltoutenc.dtx v1.8d	
\endtrivlist: Correct typo . . . . .	659	General: (DPC) lowercase external file names. internal/1044 . . . . .	394
1996-10-27 ltoutenc.dtx v1.8c		1996-11-20 fontdef.dtx v2.2p	
\@strip@args: Removed macro . . . . .	358	General: lowercase fd and enc.def file names /1044 . . . . .	543
General: Added \r A . . . . .	368	1996-11-20 ltvers.dtx v1.0f	
Added		General: Check for old format modified /2319 . . . . .	39
\textasteriskcentered . . . . .	364, 375	1996-11-23 ltoutenc.dtx v1.8e	
Corrected syntax descriptions . . . . .	352	General: Corrected description . . . . .	352
Removed \aa and \AA . . . . .	364, 368, 371	Extended description . . . . .	353
1996-10-28 lplain.dtx v1.1u		1996-11-28 ltvers.dtx v1.0g	
General: (CAR) More doc changes . . . . .	15	General: Check for old format modified /2319 . . . . .	39
\dotfill: Removed math mode . . . . .	34	1996-12-06 ltdirchk.dtx v1.0u	
1996-10-29 lplain.dtx v1.1v		\IfFileExists: *** removed from various messages for GNU Make. internal/2338 . . . . .	10
\dotfill: Got arithmetic correct (CAR) . . . . .	34	1996-12-06 ltfloor.dtx v1.1r	
1996-10-29 lspace.dtx v1.2u		\caption: Call \setminpage if needed. latex/2318 . . . . .	759
\gnewline: Added macro . . . . .	316	1996-12-06 ltfsini.dtx v3.0h	
\@no@lnbk: Macro replaces \lnbk and \@nolnbk . . . . .	314	General: (DPC) Remove *** from messages internal/2338 . . . . .	541
\\: Corrected and rationalised code . . . . .	314	1996-12-17 ltdefns.dtx v1.0w	
\nolinebreak: Reimplemented both using \no@lnbk . . . . .	313	\g@addto@macro: Use \begingroup to save making a mathord . . . . .	107
1996-10-31 ltfinal.dtx v1.0z		1996-12-20 ltsect.dtx v1.0z	
General: Added extra \lcode, hoping it does no harm in T1 (pr/1969) . . . . .	980, 987	\@dottedtocline: Added \nobreak for latex/2343 . . . . .	754
1996-10-31 ltlists.dtx v1.0p		1997-01-08 fontdef.dtx v2.2q	
\@trivlist: Added check for missing item in outer list . . . . .	658	General: Use \DeclareMathDelimiter to set delimiter codes . . . . .	552
1996-10-31 ltsect.dtx v1.0y		\mathparagraph: Define using \DeclareMathSymbol . . . . .	561
General: Corrected and tidied documentation; removed long lines . . . . .	743	1997-01-08 ltfiles.dtx v1.1j	
1996-11-03 lplain.dtx v1.1w		\cinclude: reset \deadcycles latex/2365 . . . . .	341
\dotfill: Saved tokens by using \hb@xt@ . . . . .	34	1997-01-08 ltmath.dtx v1.1d	
1996-11-04 lterror.dtx v1.2m		\root: (DPC) Remove spurious space tokens from plain TeX definition /2359 . . . . .	637
\@nodocument: Always define \@nodocument in kernel, so that it can be cleared by \document. . . . .	286	1997-02-05 ltdefns.dtx v1.0x	
1996-11-04 ltlists.dtx v1.0q		\g@addto@macro: missing percent /2402 . . . . .	107
\@trivlist: Moved check for missing item: only checked when not inlabel flag is false . . . . .	658	1997-02-21 ltlists.dtx v1.0r	
1996-11-05 ltfiles.dtx v1.1i		\@item: \ifvoid check added for \noindent. latex/2414 . . . . .	662
\nofiles: Standard \if@nobreak test added . . . . .	337	1997-03-21 ltcounts.dtx v1.1e	
1996-11-09 ltmath.dtx v1.1c		\fnsymbol: Use \mathsection and \mathparagraph. latex/2445 . . . . .	402
\@ensuredmath: Made long, as it was before. /2104 . . . . .	647		
1996-11-18 ltfsbas.dtx v3.0s			
\define@newfont: (DPC) lowercase fd file names. internal/1044 . . . . .	421		

1997-04-14 ltfiles.dtx v1.1k		1997-08-29 ltoutenc.dtx v1.9f	
\document: Set the document space factor defaults. latex/2404 . . . . .	333	General: Added OT4 encoding, provided by Marcin Woliński. . . . .	351
\normalsfcodes: Macro added (from patch file) latex/2404 . . . . .	337		
1997-04-14 ltoutput.dtx v1.2b		1997-09-09 ltdefns.dtx v1.2z	
\@writesetup: Call \normalsfcodes (from patch file) latex/2404 . . . . .	927	\provide@command: Use \begingroup to avoid generating math ords if used in math mode. pr/2573 . . . . .	86
Move \label and \index (from patch file) . . . . .	927		
1997-04-24 ltbibl.dtx v1.1m		1997-09-15 ltpictur.dtx v1.1g	
\@citex: \@empty to avoid primitive error on empty cite keys. latex/2432 . . . . .	784	\@getcirc: Warn if lines become invisible pr/2524 . . . . .	728
1997-04-30 ltoutenc.dtx v1.9a		\@picture@warn: Macro added pr/2524 . . . . .	728
General: Changed \textsc to \scshape . . . . .	365	\@sline: Warn if lines become invisible pr/2524 . . . . .	717
Introduced \textcopyright and modified \copyright . . . . .	365		
Introduced \textcopyright and modify \copyright . . . . .	366		
Modified \textunderscore, removing \mathunderscore . . . . .	365	1997-10-06 lccounts.dtx v1.1f	
Modified \underscore, removing \mathunderscore . . . . .	366	\@Roman: Change \@Roman to be fully expandable, so that the result is written properly to files. . . . .	402
1997-04-30 ltoutenc.dtx v1.9b		\@slowromancap: Macro added. . . . .	402
General: Added \leavevmode to \textunderscore . . . . .	365		
1997-05-04 ltoutenc.dtx v1.9c		1997-10-08 ltlogos.dtx v1.1h	
General: Added ‘hex index tabs’ . . . . .	372	\LaTeX: Simplify macro (force loading of suitable math fonts once). . . . .	329
Added TS1 encoding v2.2.beta . . . . .	378		
1997-05-07 ltoutenc.dtx v1.9d		1997-10-10 ltclass.dtx v1.0y	
General: Added \leavevmode to \textcompwordmark . . . . .	365	\endfilecontents: \@currenvir in banner . . . . .	826
1997-05-07 ltspace.dtx v1.2v		\reserved@c not \verb@out to save a csname . . . . .	826
\newline: Made completely robust. . . . .	315	Check for text before or after \end environment. latex/2636 . . . . .	827
1997-05-29 ltfsstrc.dtx v3.0j		Use \@gobbletwo . . . . .	826
General: Replaced \\ by \MessageBreak, as suggested by Donald Arseneau. . . . .	457		
1997-05-29 ltlogos.dtx v1.1f		1997-10-17 lfntcmd.dtx v3.3w	
\LaTeXe: Added \math so that the L <sup>A</sup> T <sub>E</sub> X 2 <sub><math>\epsilon</math></sub> logo works with non-zero values of \mathsurround. . . . .	329	\check@nocorr@: Check for vertical mode moved here, from \DeclareTextFontCommand (see PR/2646). . . . .	571
1997-06-16 ltdirchk.dtx v1.0v		\DeclareTextFontCommand:	
General: documentation improvements mainly from internal/2520 . . . . .	1	Reinstalled \check@icr as check is now done in \check@nocorr@ (see PR/2646). . . . .	569
1997-06-16 ltfloat.dtx v1.1s			
General: documentation fixes . . . . .	756	1997-10-20 ltfinal.dtx v1.1a	
1997-06-16 lfntcmd.dtx v3.3v		\cuclclist: Removed \aa and \AA from \cuclclist as these are macros. . . . .	987
General: Fix typo in documentation. . . . .	567		
1997-08-05 ltoutenc.dtx v1.9e		1997-10-21 ltdefns.dtx v1.2z1	
General: Corrected order of arguments in \UseTextSymbol example. . . . .	352	\renew@command: Use \begingroup/\endgroup rather than braces for grouping, to avoid generating empty math atom. . . . .	84

\nfss@catcodes: Moved \makeatletter from \try@load@font@shape. . . . .	421	Removed default settings, see next section. . . . .	378
1997-11-09 ltoutput.dtx v1.2c \@specialoutput: Remove incorrect code: only one \emptycol is needed here . . . . .	916	1997-12-19 ltoutenc.dtx v1.9i General: Documentation corrections. 351	
\@topnewpage: Documentation of vsize check enhanced . . . . .	913	1997-12-20 fontdef.dtx v2.2s General: Added documentation . . . . .	545
1997-11-13 ltfsdcl.dtx v3.0f \DeclareSymbolFont: (DPC) Really update \group@list don't leave new version in \toks@. latex/2661 497		1997-12-31 ltoutenc.dtx v1.9k General: Further correction . . . . .	352
\stepcounter: (DPC) Remove as never used. (Re)defined in ltcounts . . . . .	487	1998-01-12 ltoutenc.dtx v1.9k General: Added \ProvidesPackage for textcomp.sty . . . . .	351
1997-11-19 ltfloat.dtx v1.1t \@footnotetext: Missing percent, again . . . . .	775	Adding missing braces and \ushape. . . . .	380
1997-11-19 ltoutput.dtx v1.2d \@vtryfc: Reindent code, to be understandable(DPC). . . . .	933	1998-01-16 ltoutenc.dtx v1.9m General: fixed decimal codes. latex/2734 . . . . .	376
1997-11-20 ltfsdcl.dtx v3.0g \document@select@group: (DPC) inline use of \stepcounter (faster, and saves a csname per math version as no reset list) . . . . .	490	1998-03-04 ltdefns.dtx v1.2z2 \@xargdef: Unnecessary \expandafter removed: pr/2758 . 82	
\select@group: (DPC) inline use of \stepcounter (faster, and saves a csname per math version as no reset list) . . . . .	488	1998-03-05 ltoutenc.dtx v1.9n General: Added masc/fem ords as in pr/2579 . . . . .	365
1997-11-23 ltoutenc.dtx v1.9g General: Use \textperthousand, \textpertenthousand and \textfractionsolidus not \textpermill, \textperenmill and \textfraction. /2673 . . . . .	378	1998-03-20 ltdefns.dtx v1.2z3 \@thirdofthree: Macro added . . . . .	87
1997-12-17 ltoutenc.dtx v1.9h General: Added \textperthousand and \textpertenthousand . 369, 370		1998-03-20 ltoutenc.dtx v1.9o General: Documentation added about order of decls . . . . .	354
Added code for textcomp.sty. . . . .	394	Documentation added for pr/2783 353	
Added section. . . . .	394	\UndeclareTextCommand: Macro added for pr/2783 . . . . .	362
Added textcomp.sty. . . . .	351	1998-03-20 lttextcomp.dtx v1.9o General: Added various	
As in OT1, Added \leavevmode at start of \c, otherwise the output routine might be invoked within the macro. . . . .	370	\UndeclareTextCommand declarations for pr/2783 . . . . .	605
Changed to decimal codes in \ooalign. . . . .	380	Load decls after defaults for speed. 605	
Changed to decimal codes. . . . .	376	1998-03-21 ltclass.dtx v1.0z General: Added to documentation of	
Documentation changes and additions. . . . .	351	filecontents . . . . .	791
Example corrected, braces removed. . . . .	351	1998-03-21 ltclass.dtx v1.1a \@providesfile: Allow & Internal/2702 . . . . .	805
		General: Correct to new onlypreamble command list . . . . .	840
		1998-03-25 ltfsbas.dtx v3.0u \showhyphens: Suppress unnecessary error when used in preamble . . . . .	430
		1998-04-11 fontdef.dtx v2.2t General: Added \mathring accent (pr2785) . . . . .	559
		1998-04-15 fontdef.dtx v2.2u General: Use new syntax for \DeclareMathDelimiter . . . . .	552

1998-04-15 ltfssdcl.dtx v3.0h		1998-08-17 ltdirchk.dtx v1.0w
\@xxDeclareMathDelimiter: Macro added (pr/2662) . . . . .	507	General: (RmS) Documentation improvements. . . . .
1998-04-17 fontdef.dtx v2.2v		1998-08-17 lfntcmd.dtx v3.3x
General: Reinsert symbol defs for < and > chars. . . . .	553	General: (RmS) Minor documentation fixes. . . . .
1998-04-18 fontdef.dtx v2.2w		1998-08-17 lfssbas.dtx v3.0v
General: Reinsert symbol def for / char. . . . .	553	General: (RmS) Documentation fixes. .
1998-05-07 ltclass.dtx v1.1b		1998-08-17 lfssdcl.dtx v3.0i
\@onefilewithoptions@clashchk: Modify help message for latex/2805 . . . . .	818	General: (RmS) Corrected minor glitches in changes entries. . . . .
1998-05-18 ltab.dtx v1.1j		1998-08-17 lfssini.dtx v3.0i
\@endpbox: Use \setlength to set \hsize, so that the changes in the calc package apply here. . . . .	705	General: (RmS) Minor documentation fixes. . . . .
\@tabular*: Use \setlength, so that calc extensions apply. . . . .	694	1998-08-17 ltlogos.dtx v1.1i
1998-05-20 ltfinal.dtx v1.1b		General: (RmS) Minor documentation fixes. . . . .
General: Set up lccodes before loading hyphenation files: pr/2639 . . . . .	979	1998-08-17 ltmiscen.dtx v1.1g
Set up uc/lccodes after loading hyphenation files: pr/2639 . . . . .	986	General: (RmS) Minor documentation fixes. . . . .
1998-05-28 lterror.dtx v1.2n		1998-08-17 ltspace.dtx v1.2w
\@notdefinable: Added message re ‘end..’ pr/1555 . . . . .	286	General: Documentation fixes. . . . .
1998-06-04 ltboxes.dtx v1.1c		1998-08-17 preload.dtx v2.1g
\@rule: Support calc-expressions . .	679	General: (RmS) Minor documentation fixes. . . . .
1998-06-12 ltoutenc.dtx v1.9p		1998-09-19 ltoutenc.dtx v1.9r
General: Corrected 130 and 131, see pr/2834 . . . . .	381	\@a: Added \string (pr/2878) . . . . .
Renamed \textmacron pr/2840 . .	382	1998-11-13 ltab.dtx v1.1m
1998-06-12 ltoutenc.dtx v1.9q		\@array: Check for hmode to see if something went wrong during parsing (pr/2884) . . . . .
\add@accent: Explicitly set \spacefactor after \accent (pr/2877) . . . . .	358	1999-01-05 fontdef.dtx v2.2x
1998-06-12 lttextcomp.dtx v1.9p		General: Need special protection for character > in \changes entry. . . . .
General: Renamed \textmacron pr/2840 . . . . .	601	1999-01-06 lfssbas.dtx v3.0w
1998-06-18 ltab.dtx v1.1k		\@DeclareFontEncoding: Added \LastDeclaredEncoding to support cyrillic integration (pr/2988) . . . . .
General: Small addition to documentation . . . . .	682	\@LastDeclaredEncoding: Added \LastDeclaredEncoding to support cyrillic integration (pr/2988) . . . . .
1998-07-06 ltab.dtx v1.1l		1999-01-06 ltoutenc.dtx v1.9r
General: Small correction to documentation . . . . .	682	\@strip@args: New impl for latex/2930 . . . . .
1998-08-17 ltboxes.dtx v1.1e		General: Minor documentation fix. . . . .
General: (RmS) Minor Documentation fixes. . . . .	666	1999-01-06 ltoutput.dtx v1.2e
1998-08-17 ltclass.dtx v1.1c		\@makecol: Added negative vskip, as when processing outputbox below:
General: (RmS) Minor documentation fixes. . . . .	791	

suggested by Fred Bartlett pr/2892 . . . . .	922	New macro added . . . . .	83
1999-01-07 ltdefns.dtx v1.3a \@ifnextchar: made long . . . . .	102	1999-06-10 ltoutenc.dtx v1.9u General: Ensure that we also forget old options (pr/2888) . . . . .	396
\@newenvb: made long and brace optional arg. latex/2896 . . . . .	85	1999-06-12 ltoutenc.dtx v1.9v General: Extend \@uclclist only once . . . . .	395
\@testopt: made long and brace optional arg. latex/2896 . . . . .	82	1999-10-09 ltmath.dtx v1.1e \active@math@prime: Macro added, see PR 3104. . . . .	641
1999-01-07 ltdefns.dtx v1.3b \@ifnextchar: extra \long. latex/2902 . . . . .	102	\prime@s: Introduce \active@math@prime. . . . .	642
1999-01-07 ltoutenc.dtx v1.9r General: Hackery to allow using fontenc several times . . . . .	396	1999-10-09 ltoutput.dtx 1.2f \@activechar@info: Reset definition of active prime character (used in math mode) . . . . .	925
Hackery to temp support cyrillic uc/lc . . . . .	394	1999-10-28 ltoutenc.dtx v1.9w \add@accent: Give \accent@spacefactor a default definition (pr/3084) . . . . .	358
1999-01-13 ltoutenc.dtx v1.9s \@strip@args: Simplified solution for latex/2930 . . . . .	360	1999-12-08 ltoutenc.dtx v1.9x General: Changed \CYRRHOOK and \cyrrhook to \CYRRHK and \cyrrhk as name changed in the cyrillic bundle for naming consistency with other "hook" glyphs. . . . .	394
1999-01-18 ltdefns.dtx v1.3c \@yargd@f: New implementation DPC /2942 . . . . .	83	2000-01-07 ltmiscen.dtx v1.1h \@verbatim: Disable hyphenation even if the font allows it. . . . .	630
1999-02-09 ltdefns.dtx v1.3d \@yargd@f: catch bad argument forms by re-inserting #3 . . . . .	83	2000-01-15 ltpictur.dtx v1.1i \@upvector: Removed space at end-of-line, CAR . . . . .	720
1999-02-12 lttextcomp.dtx v3.0j \legacyoldstylenums: Use \rmdefault instead of cmm (pr/2954) . . . . .	576	2000-01-30 ltfntcmd.dtx v3.3y \DeclareTextFontCommand: Use \hmode@bgroup now (pr/3160) . . .	569
1999-02-24 ltoutenc.dtx v1.9t General: Corrected hackery cyrillic uc/lc list . . . . .	394	2000-01-30 ltoutenc.dtx v1.9y General: Use \hmode@bgroup where applicable (pr/3160) . . . . .	367–370, 375–378, 380
1999-03-01 ltdefns.dtx v1.3e \@ifnextchar: remove extra \long. internal/2967 . . . . .	102	\add@accent: Use \hmode@bgroup where applicable (pr/3160) . . . . .	357
1999-04-15 ltpictur.dtx v1.1h \@getlarrow: Replaced octal number, CAR . . . . .	719	\hmode@bgroup: Macro added . . . . .	358
\@upvector: Replaced octal number, CAR . . . . .	720	2000-01-30 ltoutenc.dtx v1.9z \@use@text@encoding: Macro reimplemented (pr/3160) . . . . .	360, 361
General: Replaced octal number, CAR . . . . .	719, 720	\add@accent: Macro reimplemented (pr/3160) . . . . .	357
Replaced octal numbers, CAR . . . . .	707	\hmode@start@before@group: Macro added (pr/3160) . . . . .	361
1999-04-19 ltfloat.dtx v1.1u \caption: Made caption an error outside a float: latex/2815 . . . . .	759	2000-05-19 ltmiscen.dtx v1.1i \enddocument: Reset \AtEndDocument for latex/3060 . . . . .	615
1999-04-27 ltboxes.dtx v1.1f \@parboxto: (CAR) Changed \empty to \relax as flag for natural width: pr/2975 . . . . .	675		
1999-04-29 ltdefns.dtx v1.3f \@yargd@f: Full expansion and conversion needed for digit in new version, see pr/3013 . . . . .	83		

2000-05-26 ltpage.dtx v1.0j		2001-02-16 ltxref.dtx v1.1k	
\@markright: Reimplementation to fix expansion error (pr/3203). . . . .	789	\@newl@bel: Added an extra grouplevel (PR3250), jlb . . . . .	611
\leftmark: Use \empty instead of brace group (pr/3203). . . . .	789	\@providesfile: Explicitly set catcode of \endlinechar to 10 (pr/3334) . . . . .	805
\markright: Reimplementation to fix expansion error (pr/3203). . . . .	788	2001-05-25 ltclass.dtx v1.1d	
\rightmark: Use \empty instead of brace group (pr/3203). . . . .	789	\@providesfile: Explicitly set catcode of \endlinechar to 10 (pr/3334) . . . . .	805
2000-06-02 ltpage.dtx v1.0k		2001-05-25 ltdirchk.dtx v1.0x	
\@markright: Small adjustment to give slightly less expansion, CAR	789	General: Explicitly set catcode of \endlinechar to 10 (pr/3334) . . .	4
\markright: Small adjustment to give slightly less expansion, CAR . . .	788	2001-05-28 ltoutenc.dtx v1.93	
Tidied 1.0j reimplementation, CAR	788	General: Added composites for compatibility with T1, pr/3295 . . .	369
2000-07-11 ltmiscen.dtx v1.1j		Changed the effect of \.\i, pr/3295 . . . . .	372
\@enddocument@kernel@warnings: Fix typo in warning . . . . .	616	2001-06-02 fontdef.dtx v2.2y	
2000-07-12 ltoutput.dtx 1.2g		General: Provide default cfg files (pr/3264) . . . . .	562
General: Ensure that rule is in \normalcolor . . . . .	966	2001-06-04 fontdef.dtx v2.2z	
2000-07-12 ltoutput.dtx 1.2i		General: Guard against math active equal and pipe sign in \models (pr/3333) . . . . .	558
\@makecol: Removed negative vskip, as it gives unacceptable results when the depth is large: pr/3189	922	Guard against math active equal sign in \Relbar (pr/3333) . . . . .	558
2000-07-19 ltoutput.dtx v1.2h		2001-06-04 ltclass.dtx v1.1e	
\@writesetup: Reset and restore \@if@newlist for internal/3231 .	926	\@providesfile: But only if it is a char (pr/3334) . . . . .	805
2000-08-23 ltfinal.dtx v1.1c		2001-06-04 ltdirchk.dtx v1.0y	
General: Fix typo in warning . . . . .	981	General: But only if it is a char (pr/3334) . . . . .	4
2000-08-30 ltoutenc.dtx v1.91		2001-06-04 ltpictur.dtx v1.1j	
\@use@text@encoding: Rearranged but no change to final code, CAR (pr/3160) . . . . .	360	\@sline: Don't warn for exactly zero pr/3318 . . . . .	717
\add@accent: Rearranged but no change to final code, CAR (pr/3160) . . . . .	357	2001-06-04 ltvers.dtx v1.0i	
2000-09-01 ltfinal.dtx v1.1d		General: Check for old format disabled	39
\errhelp: Set error help empty at very end (pr/449 done correctly).	989	2001-06-05 ltoutenc.dtx v1.94	
2000-09-24 ltfloat.dtx v1.2b		General: Text composite Commands need kludges for ',' – see tbl1903.lvt . . . . .	369
\end@dblfloat: FMi: use output routine to defer float . . . . .	764	2001-08-26 ltclass.dtx v1.1f	
2000-09-24 ltoutput.dtx v1.2b		\@providesfile: Readded setting of space char (pr/3353) . . . . .	805
\@doclearpage: FMi: ensure \doclearpage is called again until all floats are output. . . . .	920	2002-02-24 lplain.dtx v1.1x	
2000-09-24 ltoutput.dtx v1.2n		\loggingall: Macro added . . . . .	35
\@addtocurcol: FMi: test for wide float was in wrong place . . . . .	938	\loggingoutput: Macro added . . . . .	34
2001-01-07 ltoutput.dtx v1.2j		\showoutput: Use newly added \loggingoutput . . . . .	34
\@writesetup: And do it in the right macro (pr/3286) . . . . .	926	\tracingall: Use newly added \loggingoutput . . . . .	35
		2002-06-16 ltoutenc.dtx v1.95	
		General: Added \textbardbl (pr/3400) . . . . .	375
		Added default for \textbardbl (pr/3400) . . . . .	364

2002-06-17 ltoutenc.dtx v1.95		2004-01-03 ltoutenc.dtx v1.99b	
General: Corrected \c for T1 (pr/3442) .....	370	General: Added \textogonekcentered (pr/3532) .....	370
Definition of \texttexclamdown changed (pr/3368) .....	368	Added composites for \k (pr/3532) ..	375
Definition of \textquestiondown changed (pr/3368) .....	368	Use \oalign for \k (pr/3532) ..	370
2002-06-18 ltoutenc.dtx v1.95		2004-01-04 ltbibl.dtx v1.1p	
General: Changed def for \textregistered to avoid small caps (pr/3420) .....	365	\nocite: Changed error message ..	785
2002-10-01 ltfloat.dtx v1.1v		2004-01-04 ltoutenc.dtx v1.99c	
\thempfootnote: Use braces around \itshape to keep font change local (pr/3460) .....	773	General: More adjustments for ogonek (pr/3532) .....	370
2002-10-02 ltfssbas.dtx v3.0x		2004-01-23 ltdefns.dtx v1.1g	
\DeclareFontSubstitution: Adding \LastDeclaredEncoding introduced a bug as on some occasions that macro name was stored in the internal lists instead of the actual encoding. (pr/3459)	411	\newenva: Use kernel version of \ifnextchar (pr/3501) .....	85
2002-10-28 ltlists.dtx v1.0s		\@testopt: Use kernel version of \ifnextchar (pr/3501) .....	82
\endtrivlist: Check for math mode (pr/3437) .....	659	\@xargdef: Use kernel version of \ifnextchar (pr/3501) .....	82
2002-10-28 ltoutenc.dtx v1.96		\@xdblarg: Use kernel version of \ifnextchar (pr/3501) .....	104
General: coding change, to follow bug fix by DEK in plain.tex (pr/3469) .....	368, 377	2004-01-23 ltdefns.dtx v1.3g	
2002-12-13 ltbibl.dtx v1.1n		\kernel@ifnextchar: Added macro (pr/3501) .....	103
\@citex: Added \leavevmode in case citation is at start of paragraph (pr/3486) .....	784	2004-01-28 ltclass.dtx v1.1g	
2003-01-01 ltfntcmd.dtx v3.3z		\@providesfile: Use kernel version of \ifnextchar (pr/3501) .....	805
General: Code checked and documentation extended by Chris	569	2004-01-28 ltvers.dtx v1.0k	
2003-05-18 ltbibl.dtx v1.1o		General: Check for old format made 5 years (pr/3601) .....	39
\nocite: Check if we are after \document .....	785	2004-02-02 fontdef.dtx v2.3	
2003-08-27 ltpictur.dtx v1.1k		General: Many things from here on made robust .....	557
\@bezier: added missing displacement pr/3566 .....	736	2004-02-02 ltoutenc.dtx v1.99	
\@sline: check for \clinechar being empty pr/3570 .....	717	General: Added \textbigcircle ..	375
2003-10-13 ltfinal.dtx v1.1e		2004-02-04 fontdef.dtx v2.3a	
General: Added extra \lccode for \-		General: Added bigtriangle synonyms for stmaryrd .....	555
and \textcompwordmark .....	980	2004-02-04 ltspace.dtx v1.3	
2003-12-16 ltoutput.dtx v1.2k		\nobreakdashes: (Macro added ...)	325
\@makecol: Ensure that \elt has a defined state (pr/3586) .....	922	2004-02-06 ltoutenc.dtx v1.99d	
2003-12-30 ltpictur.dtx v1.1j		\@cinmathwarn: New command added to fix severe bug: pr/3563 .....	355
\@getcirc: issue warning if circle size can't be met pr/3473 .....	728	2004-02-07 ltoutput.dtx v1.2l	
		\@doclearpage: Empty kludgeins box if necessary, pr/3528 .....	920
		2004-02-13 ltoutenc.dtx v1.99e	
		General: Documentation fixes: typos	351
		2004-02-15 ltbibl.dtx v1.1q	
		\@cite@ofmt: Added hook with default value \hbox .....	786
		\@citex: Changed to use a hook with default value \hbox .....	784
		2004-02-15 ltspace.dtx v1.3a	
		\nobreakdashes: (Added spacefactor setting .....	325

2004-10-20 ltoutput.dtx v1.2m		2009-12-14 ltfntcmd.dtx v3.4a	
\@makecol: Removed dead code . . .	922	\ifmaybe@ic: Macro added . . . . .	572
2005-07-27 ltfssdcl.dtx v3.0j		\maybe@ic@: Use switch \ifmaybe@ic instead of \if@tempswa . . . . .	572
\DeclareMathAlphabet: (MH) Make document commands robust . . .	499	\t@st@ic: Use switch \ifmaybe@ic instead of \if@tempswa . . . . .	573
\DeclareSymbolFontAlphabet: (MH) Make document commands robust	512		
\new@mathalphabet: (MH) Make document commands robust . . .	500	2010-08-17 ltmiscen.dtx v1.1k	
\non@alphe rr: (MH) Change because command is now properly robust	489	\enddocument: Use braces around \input arg (pr/4124) . . . . .	616
\SetMathAlphabet: (MH) Make document commands robust . . .	501	2010-08-17 ltmiscen.dtx v1.1l	
2005-09-27 ltoutenc.dtx v1.99g		\enddocument: Change of plan: use \@@input instead (pr/4124) . . . . .	616
General: Replace \sh@ft by \ltx@sh@ft . . . . .	367, 370, 377	2011-05-08 ltfssdcl.dtx v3.0n	
2005-09-27 lplain.dtx v1.1y		\in@: Simplified thanks to Bruno. . . . .	485
\ltx@sh@ft: New macro . . . . .	34	2011-08-19 ltclass.dtx v1.1i	
\sh@ft: Macro no longer used but left for compatibility . . . . .	33	\@ifclasswith: Re-jig definition after more stringent \in@ test. . . . .	802
2005-11-08 ltoutenc.dtx v1.99h		2011-09-03 ltfssdcl.dtx v3.0o	
General: Added \ij and \IJ from babel. (pr/3771) . . . . .	364, 369, 371	\new@mathversion: (Will) Remove \global before \newcount (unnecessary and caused etex bug). . . . .	496
2005-11-10 ltmath.dtx v1.1g		2012-01-20 lplain.dtx v2.0b	
\l: (MH) Fixed potential problem in \l (pr/3399). . . . .	642	\loggingall: etex tracing if available . . . . .	35
General: (MH) Minor documentation fixes. . . . .	635	2013-07-07 ltclass.dtx v1.1i	
2006-05-18 ltboxes.dtx v1.1g		General: Correctly describe how the date in \ifpackage@later is used . . . . .	794
\@parboxto: Ensure \@parboxto holds the value of \tempdimb not the register itself (pr/3867) . . . . .	675	2014-04-18 ltoutput.dtx v1.1o	
2006-09-13 ltoutput.dtx v1.1m		General: Handle infinite glue from \enlarge@thispage (pr/4023) . . . . .	967
General: Ensure that rule is in \normalcolor . . . . .	967	2014-04-24 ltoutput.dtx v1.2n	
2007-08-05 ltclass.dtx v1.1h		\f@trace@message: Renamed internal trace commands; provide as package . . . . .	956
\@fileswithoptions: Prevent loss of brackets PR/3965 . . . . .	814	2014-04-27 ltfloat.dtx v1.2b	
2007-08-06 ltcntrl.dtx v1.0h		\end@dblfloat: Inline the code to allow some coexistence with packages that hook into \end@float and do not know about the algorithm change . . . . .	764
\@fornoop: Really make defs long . . . . .	278	2014-06-10 ltfloat.dtx v1.2b	
2007-08-31 ltfssdcl.dtx v3.0l		\end@dblfloat: missing \fi added . . . . .	764
\SetSymbolFont@: Font warning changed to info for encoding change (pr/3975) . . . . .	498	2014-12-30 ltfinal.dtx v2.0a	
2009-09-24 ltvers.dtx v1.0l		\newmarks: macro added . . . . .	974
General: Stop checking for old format . . . . .	39	\newXeTeXintercharclass: macro added . . . . .	974
2009-10-20 ltfssdcl.dtx v3.0m		2014-12-30 ltffloat.dtx v1.2a	
\in@: More robust thanks to Heiko. . . . .	485	\@textsubscript: Command added (latexrelease) . . . . .	775
2009-10-28 lttextcomp.dtx v1.99k		\textsubscript: Command added (latexrelease) . . . . .	774
General: Added Latin Modern and TeX Gyre subsets . . . . .	606	2014-12-30 ltfssbas.dtx v3.0y	
2009-11-04 lttextcomp.dtx v1.99l		\mathgroup: move allocation to lplain. . . . .	407
General: Added more Latin Modern and TeX Gyre subsets . . . . .	606		

2014-12-30 ltoutput.dtx v1.2m		2015-01-10 lccounts.dtx v1.1h	
General: Command updated (latexrelease) . . . . .	966	\@fnnsymbol: Unse \TextOrMath (latexrelease) . . . . .	403
2014-12-30 lplain.dtx v2.0a		\@stpelt: Reset all within counters in one go (latexrelease) . . . . .	399
\@alloc: macro added . . . . .	20	2015-01-11 lccounts.dtx v1.1h	
\@alloc@chardef: macro added . . . . .	19	\TextOrMath: Add command to solve robustness issues (pr/3752) (latexrelease) . . . . .	404
\@alloc@top: macro added . . . . .	19	2015-01-11 lffloat.dtx v1.2b	
\@ch@ck: macro added . . . . .	20	\@dblfloatplacement: float order in 2-column (latexrelease) . . . . .	766
\@extrafloats: macro added . . . . .	21	\@xfloat: Check for valid option (latexrelease) . . . . .	760
\newlanguage: New engine-specific allocation scheme (latexrelease) . . . . .	18	\end@dblfloat: float order in 2-column (latexrelease) . . . . .	764
2014-12-30 ltspace.dtx v1.3b		2015-01-11 lfssbas.dtx v3.0y	
\@: \@ discards spaces when moving (pr3039)(latexrelease) . . . . .	326	\@DeclarMathSizes: Allow arbitrary units (latexrelease) . . . . .	414
2015-01-03 ltdirchk.dtx v1.4a		2015-01-11 ltspace.dtx v1.3d	
\typein: use modified definition in luatex . . . . .	80	\@Eshack: Allow hyphenation (Donald Arseneau pr/3498) (latexrelease) . . . . .	319
2015-01-03 ltdirchk.dtx v1.1		\@esphack: Allow hyphenation (Donald Arseneau pr/3498) (latexrelease) . . . . .	318
General: Enable extra primitives when LuaTeX is used . . . . .	3	2015-01-14 ltoutput.dtx v1.2n	
2015-01-03 ltfinal.dtx v2.0a		\@addtocurcol: float order in 2-column (latexrelease) . . . . .	937
General: Skip resetting codes with Unicode engines . . . . .	986	\@addtobdblcol: float order in 2-column (latexrelease) . . . . .	948
Unicode data loading added . . . . .	977	\@adtonextcol: float order in 2-column (latexrelease) . . . . .	944
2015-01-07 ltvers.dtx v1.0n		\@doclearpage: Empty kludgeins box if necessary, pr/3528 . . . . .	919
\@check@IncludeInRelease: macro added . . . . .	41	\@startdblcolumn: float order in 2-column (latexrelease) . . . . .	931
2015-01-08 ltboxes.dtx v1.1h		\@xtryfc: float order in 2-column (latexrelease) . . . . .	933
\framebox: Make Robust (latexrelease) . . . . .	673	\@ztryfc: float order in 2-column (latexrelease) . . . . .	934
\makebox: Make Robust (latexrelease)	667	2015-01-14 ltspace.dtx v1.3e	
\parbox: Make Robust (latexrelease)	674	\@addpenalty: Avoid adding redundant skips (DPC) . . . . .	322
\raisebox: Make Robust (latexrelease) . . . . .	680	2015-01-17 ltvers.dtx v1.0m	
\rule: Make Robust (latexrelease) . . . . .	679	\@check@IncludeInRelease: modified with \@currname . . . . .	41
\savebox: Make Robust (latexrelease)	670	2015-01-19 ltvers.dtx v1.0o	
2015-01-08 ltdefns.dtx v1.4a		\@check@IncludeInRelease: Optional argument . . . . .	41
\MakeRobust: Added macro . . . . .	90		
2015-01-08 ltlength.dtx v1.1c			
\setlength: to ensure first length argument is terminated. (latexrelease) . . . . .	405		
2015-01-08 ltmath.dtx v1.1h			
\@: Make Robust (latexrelease) . . . . .	642		
\]: Make Robust (latexrelease) . . . . .	642		
2015-01-09 lfssini.dtx v3.1a			
\em: Allow \emph to produce small caps (latexrelease) . . . . .	535		
\emminershape: macro added (latexrelease) . . . . .	535		
2015-01-09 ltspace.dtx v1.1h			
\addpenalty: Donald Arseneau's fix from PR/377703 (latexrelease) . . . . .	322		

2015-01-20 ltoutput.dtx v1.2m		2015-02-21 ltpictur.dtx v1.1k
\f@tracemessage: Reset		General: Removed autoload code ... 707
\IncludeInRelease flags ..... 958		
2015-01-22 ltvers.dtx v1.0p		2015-02-21 lplain.dtx v2.0e
General: Preserve any \everyjob		General: Removed autoload code ... 15
material inserted by a loader (.ini		
file) ..... 40		
2015-01-23 ltfinal.dtx v2.0b		2015-02-21 ltab.dtx v1.1n
\newmarks: use reserved count 256 . 974		General: Removed autoload code ... 682
\newXeTeXintercharclass: use		
reserved count 257 ..... 974		
2015-01-23 lplain.dtx v2.0c		2015-02-21 ltvers.dtx v1.0r
\xtrafloats: reserve counts 256–265 21		General: Removed autoload code ... 39
2015-01-24 ltfinal.dtx v2.0c		2015-02-21 ltvers.dtx v1.0w
General: Skip T1-code entirely with		
Unicode engines ..... 977		
2015-02-03 ltfinal.dtx v2.0d		\@check@IncludeInRelease: set
General: Set \lccode for - with		\@currname empty here (in case
Unicode engines ..... 979		\IncludeInRelease input early) . 40
2015-02-16 ltoutenc.dtx v1.99m		2015-02-22 ltfsscmp.dtx v3.0e
General: Added \textcommabelow		General: Moved all code into
latex/4414 ..... 366		latexrelease - obsolete commands
2015-02-16 ltoutenc.dtx v1.99n		are no longer automatically part of
General: Added \textcommaabove .. 367		the kernel ..... 480
Added composites for ç ..... 375		
Added composites for \c ..... 369		
2015-02-16 lttextcomp.dtx v1.99m		2015-03-02 lplain.dtx v2.0f
General: Added lmtt (Heiko Oberdiek)		\@mathgroup@top: macro added ... 20
latex/4415 ..... 606		\newlanguage: allow 255 math groups
2015-02-19 ltvers.dtx v1.0q		in Unicode engines ..... 18
\@check@IncludeInRelease: Swap		2015-03-10 lplain.dtx v2.0g
argument order ..... 41		\hideoutput: macro added ..... 37
2015-02-20 lplain.dtx v2.0d		\loggingall: Reorganize to be less
\loggingall: Spell commands		noisy ..... 35
correctly :-) ..... 35		\tracingnone: macro added ..... 36
2015-02-21 ltfdfns.dtx v1.4b		2015-03-12 ltoutput.dtx v1.2m
General: Removed autoload support . 78		General: initialise \@dbldeflist
2015-02-21 lterror.dtx v1.2o		again ..... 909
General: Removed autoload support 279		2015-03-18 ltfssdcl.dtx v3.0q
2015-02-21 ltfiles.dtx v1.1m		\DeclareSymbolFont: Restrict Symbol
General: Removed autoload support 330		fonts to 0–15 ..... 497
2015-02-21 ltfssbas.dtx v3.0z		\document@select@group: Introduce
General: Removed autoload code .. 407		\@mathgroup@top ..... 490
2015-02-21 ltfsscmp.dtx v3.0d		\select@group: Introduce
General: Removed autoload code .. 480		\@mathgroup@top ..... 488
2015-02-21 ltfssdcl.dtx v3.0p		2015-03-26 ltfinal.dtx v2.0d
General: Removed autoload code .. 485		General: Use renamed
2015-02-21 lfsstrc.dtx v3.0k		unicode-letters.def ..... 977
General: Removed autoload code .. 455		2015-04-07 ltfssbas.dtx v3.1a
2015-02-21 ltoutenc.dtx v1.99m		\wrong@fontshape: Try loading fd file
General: Removed autoload code .. 351		if family has changed ..... 425
2015-02-21 ltoutput.dtx v1.2n		2015-04-28 ltfinal.dtx v2.0f
General: Removed autoload code .. 898		\newXeTeXintercharclass: define
\f@depth: macro added(latexrelease) 918		\@alloc@intercharclass for

2015-06-19 ltfinal.dtx v2.0g	\newluafunction: Macro added . . . . .	52
\@alloc@intercharclass@top: Use –1 for first range to get contiguous allocation . . . . .	\newwhatsit: Macro added . . . . .	53
975	module_error: Function added . . . . .	57
\newmarks: Use –1 for first range to get contiguous allocation . . . . .	module_info: Function added . . . . .	57
974	module_warning: Function added . . . . .	57
2015-06-19 ltplain.dtx v2.0h	modules: Function modified . . . . .	56
General: delete spurious old definition of \newtoks . . . . .	create_callback: Function added . . . . .	66
24	provides_module: Function added . . . . .	56
\@alloc: extra braces in case arguments not single token . . . . .	luatexbase: Table added . . . . .	56
20	2015-10-02 ltdirchk.dtx v1.2a	
\newlanguage: Use –1 for first range to get contiguous allocation . . . . .	General: Allow backing out of unprefixed names . . . . .	3
18	2015-10-02 ltluatex.dtx v1.0b	
2015-06-23 ltfinal.dtx v2.0h	General: Fix backing out of TeX code . . . . .	54
General: set \patch@level in ltvers rather than in ltfinal/ltpatch . . . . .	2015-10-02 ltluatex.dtx v1.0c	
988	General: Allow backing out of Lua code . . . . .	54
2015-06-23 ltvers.dtx v1.0t	2015-10-02 ltluatex.dtx v1.0e	
General: set \patch@level in ltvers rather than in ltfinal/ltpatch . . . . .	uninstall: Function added . . . . .	70
39	2015-10-03 ltluatex.dtx v1.0f	
2015-08-06 ltplain.dtx v2.0i	provides_module: use luatexbase_log . . . . .	56
\xtraffloats: Add \string in case argument is not an unexpandable primitive . . . . .	2015-10-27 ltplain.dtx v2.1b	
21	\xtraffloats: Use global assignment when switching to extended range . . . . .	21
2015-08-23 ltdirchk.dtx v1.2	2015-11-07 ltspace.dtx v1.3f	
General: Do not use luatex prefix . . . . .	\@esphack: Only space if there is no space at the end of the hlist latex/4443 . . . . .	318
3	2015-11-14 ltluatex.dtx v1.0g	
2015-08-23 ltvers.dtx v1.0v	General: Track LuaTeX changes for (new)token.create . . . . .	58
General: Allow negative patchlevel for pre-release . . . . .	2015-11-18 ltplain.dtx v2.2a	
40	\newlanguage: Extended stream allocation in luatex (0.85) . . . . .	18
2015-08-30 ltplain.dtx v2.1a	2015-11-19 ltplain.dtx v2.2b	
\newinsert: new \newinsert implementation . . . . .	\newlanguage: Only extend allocation of write streams (see luatex list) . . . . .	18
23	2015-11-27 ltluatex.dtx v1.0h	
2015-09-205 ltoutput.dtx v1.3a	\callback_descriptions: Match test in in-callback latex/4445 . . . . .	70
General: extended \freelist . . . . .	in_callback: Guard against undefined list latex/4445 . . . . .	69
908	2015-11-29 ltluatex.dtx v1.0i	
2015-09-24 ltluatex.dtx v1.0a	General: Declare this as local before used in the module error definitions (PHG) . . . . .	56
\call_callback: Function added . . . . .	\call_callback: Check name is not nil in error message (PHG) . . . . .	67
67	\create_callback: Check name is not nil in error message (PHG) . . . . .	66
\callback_register: Function modified . . . . .	2015-12-02 ltluatex.dtx v1.0j	
63	General: Adjust hashtokens to store the result of tex.hashtokens(), not the function (PHG) . . . . .	58
\callback_descriptions: Function added . . . . .		
70		
\catcodetable@atletter: Macro added . . . . .		
51		
\catcodetable@initex: Macro added . . . . .		
51		
\catcodetable@latex: Macro added . . . . .		
51		
\catcodetable@string: Macro added . . . . .		
51		
\add_to_callback: Function added . . . . .		
67		
\remove_from_callback: Function added . . . . .		
68		
\new_attribute: Function added . . . . .		
59		
\disable_callback: Function added . . . . .		
69		
\in_callback: Function added . . . . .		
69		
\newattribute: Macro added . . . . .		
50		
\newcatcodetable: Macro added . . . . .		
51		
\newluabytecode: Macro added . . . . .		
53		
\newluachunkname: Macro added . . . . .		
54		

Assorted typos fixed (PHG) . . . . .	48	2016-02-18 ltfssdcl.dtx v3.0r
Declaration/use of first_ head fixed (PHG) . . . . .	57	\@DeclareMathDelimiter: Check for delimiter not \delimiter . . . . . 508
Remove nonlocal iteration variables (PHG) . . . . .	48	\@DeclareMathAccent: Check for mathaccent not \mathaccemt . . . . . 503
Remove unreachable code after calls to error() (PHG) . . . . .	48	\@DeclareMathRadical: Check for radical not \radical . . . . . 510
2015-12-02 ltluatex.dtx v1.0k		\@DeclareMathSymbol: Check for mathchar not \mathchar . . . . . 505
General: resolve name and i.description (PHG) . . . . .	65	2016-03-13 ltluatex.dtx v1.0n
call_callback: Give more specific error messages (PHG) . . . . .	67	General: contribute_ filter added . . . . . 62
add_to_callback: Give more specific error messages (PHG) . . . . .	67	insert_ local_ par added . . . . . 62
remove_from_callback: adjust initialization of cb local (PHG) . . . . .	68	2016-03-29 ltpictur.dtx v1.11
Give more specific error messages (PHG) . . . . .	68	\@covval: add setting of line tests 729, 730
create_callback: Give more specific error messages (PHG) . . . . .	66	initialise tests . . . . . 729
2015-12-10 ltfinal.dtx v2.0i		\@covhorz: use glue not leaders if horizontal line not required . . . . . 732
General: Use new common Unicode data loaders . . . . .	977	\@covvert: use glue not leaders if vertical line not required . . . . . 731
2015-12-18 ltluatex.dtx v1.0l		\if@ovhline: macro added
General: Load Unicode data from source . . . . .	51	(latex/4452) . . . . . 729
2016-01-04 ltfinal.dtx v2.0j		\if@ovvline: macro added
General: Do not set up inter character classes for XeTeX . . . . .	977	(latex/4452) . . . . . 729
\@alloc@intercharclass@top: Start allocation at one not three . . . . .	974	2016-04-22 ltfinal.dtx v2.0q
2016-01-05 ltfinal.dtx v2.0k		\@e@alloc@intercharclass@top:
\@alloc@intercharclass@top: Remove duplicated code . . . . .	974	XeTeX 0.99996 has 4096 char classes not 256 . . . . . 975
2016-01-05 ltfinal.dtx v2.0l		2016-06-19 ltoutenc.dtx v1.99m
General: Correct latexrelease guards . . . . .	977	General: OT1 definition (was duplicate T1 definition) . . . . . 369
Ensure old definitions for inter-character class toks are available using latexrelease . . . . .	977	2016-06-20 ltclass.dtx v1.1j
Missing brace . . . . .	977	General: don't declare as
2016-01-05 ltfinal.dtx v2.0m		\@onlypreamble . . . . . 801
General: Undefine XeTeX classes when using patching an older kernel . . . . .	977	2016-07-29 lplain.dtx v2.2c
2016-01-05 ltfinal.dtx v2.0p		\extrafloats: use \global \chardef 21
General: Only apply XeTeX change if XeTeX is in use . . . . .	977	\newinsert: fix for tlb-newinsert-001 . . . . . 23
2016-02-11 ltluatex.dtx v1.0m		2016-10-02 ltclass.dtx v1.2a
General: pdf_stream_filter_callback removed . . . . .	63	\@ifclasswith: Ignore spaces while checking for option clash . . . . . 802
process_rule, [hv]pack_quality append_to_vlist_filter added . . . . .	62	\ExecuteOptions: Ignore spaces in argument . . . . . 810
read_cidmap_file added . . . . .	62	2016-10-15 ltdirchk.dtx v1.2b
show_warning_message added . . . . .	63	General: Require eT <sub>EX</sub> . . . . . 4
token_filter removed . . . . .	62	2016-10-15 lterror.dtx v1.2p
		General: Require eT <sub>EX</sub> . . . . . 279
		2016-10-15 ltfinal.dtx v2.0r
		General: Require eT <sub>EX</sub> . . . . . 974
		2016-10-15 ltfinal.dtx v2.0s
		General: Tidy up status of char 127 . . . . . 974
		2016-10-15 ltfssini.dtx v3.1b
		General: Require eT <sub>EX</sub> . . . . . 515
		2016-10-15 lplain.dtx v2.2d
		General: Require eT <sub>EX</sub> . . . . . 15

2016-10-16 ltplain.dtx v2.3a	\newlanguage: Allow languages up to 16383 in luatex . . . . .	18	2017-02-19 ltoutenc.dtx v2.0f	General: add \empty to guard against 3rd argument being empty . . . . .	368	
2016-10-19 ltcnts.dtx v1.1j	\TextOrMath: Test directly for \protected . . . . .	404	declare composites with empty base for hat and tilde, use same slots for \textasciicircum ans \textasciitilde . . . . .	383	declare straight quotes using new \remove@tlig command . . . . .	383
2016-11-06 ltplain.dtx v2.3b	General: Drop \outer entirely . . . . .	15	2017-02-22 ltoutenc.dtx v2.0g	General: Fix typo introduced at 2.0f	383	
2016-11-09 ltclass.dtx v2.1b	\@fileswithoptions: Improve \ifx tests PR/4497 . . . . .	814	2017-02-24 ltoutenc.dtx v2.0h	General: introduce		
2016-12-03 fontdef.dtx v3.0a	General: call_edit added . . . . .	63	\DeclareUnicodeAccent . . . . .	383		
2016-12-04 (DPC) Default to TU encoding for Unicode TeX engines	\shapedefault: (DPC) Default to TU encoding for Unicode TeX engines	545	\DeclareTextCompositeCommand: add check whether the accent command is defined for this encoding . . . . .	358		
2016-12-04 ltoutenc.dtx v2.0a	General: Added TU encoding . . . . .	383	2017-03-08 ltclass.dtx v1.2c	General: add \@parse@version@dash to support yyyy-mm-dd as well as yyyy/mm/dd . . . . .	801	
2017-01-01 ltoutput.dtx v1.3b	General: make fpmin negative so ignored even if float height is negative . . . . .	966	2017-03-09 ltfinal.dtx v2.0t	\l@nohyphenation: ensure \l@nohyphenation is defined. . . . .	980	
2017-01-10 ltfsbas.dtx v3.2a	\showhyphens: Add version of \showhyphens that works with XeTeX . . . . .	430	2017-03-09 ltmiscen.dtx v1.1m	\overline: Use \language not \hyphenchar . . . . .	630	
2017-01-23 ltoutenc.dtx v2.0b	General: Added TU specific commands in ASCII range pr/4500 . . . . .	383	\verb: Use \language to stop hyphenation . . . . .	634		
2017-01-24 ltoutenc.dtx v2.0c	General: Declare TU composites for i and j . . . . .	383	2017-03-10 ltfiles.dtx v1.1n	\document: Save language default . . . . .	333	
	Make \textasteriskcentered U+2217 not U+204E . . . . .	383	2017-03-10 ltoutput.dtx v1.3c	\@writesetup: Reset \language . . . . .	926	
	TeX ligature syntax for xetex and luatex reversed . . . . .	383	2017-03-13 ltdefns.dtx v1.5a	\-: Define \- in terms of \hyphenchar	105	
2017-01-24 ltoutenc.dtx v2.0d	General: Declare macron composites for YyGg . . . . .	383	2017-03-27 ltdefns.dtx v1.5b	\dischyp: Define \@dischyp after \- . . . . .	105	
2017-02-12 ltoutenc.dtx v2.0e	General: Declare fallback code for \textasteriskcentered . . . . .	383	2017-03-28 ltluatex.dtx v1.1e	General: glyph_stream_provider added . . . . .	63	
2017-02-18 ltluatex.dtx v1.1c	\new_attribute: Parameterize count used in tracking . . . . .	59	2017-03-29 ltboxes.dtx v1.3a	\arrayparboxrestore: Reset \lineskiplimit . . . . .	676	
	\new_bytocode: Parameterize count used in tracking . . . . .	60	2017-04-05 ltoutenc.dtx v2.0i	\DeclareTextCompositeCommand: Declare accent command if not already declared when declaring a composite . . . . .	358	
	\new_chunkname: Parameterize count used in tracking . . . . .	60	2017-04-10 ltplain.dtx v2.3c	\newlanguage: Correction to code to skip write18 in luatex . . . . .	18	
	\new_whatsit: Parameterize count used in tracking . . . . .	59				

2017-04-11 ltoutput.dtx v2.4a		2018-05-11 ltfinal.dtx v2.18	
\newpage: account for the depth of the last row of the page . . . . .	912	General: Make invalid UTF-8 also safe, for legacy filesystem encodings . . . . .	984
2017-12-17 ltoutput.dtx v1.4b		2018-05-29 ltclass.dtx v1.2j	
\@addtonextcol: fix doc guards . . . . .	944	\endfilecontents: use \csname not \@undefined . . . . .	827
2018-01-06 ltdefns.dtx 1.5c		2018-08-11 ltoutenc.dtx v2.0j	
\@ifundefined: Avoid defining undefined commands to \relax . . . . .	101	General: Provide \guillemetleft and \guillemetright . . . . .	371, 377, 386
2018-02-18 ltclass.dtx v1.2d		2018-08-18 ltluatex.dtx v1.1h	
\@ifl@ter: Added 0 up front to make bad data come out as 0. . . . .	801	General: append_to_vlist_ filter is exclusive . . . . .	62
General: Introduce rollback concept	832	2018-08-24 ltfinal.dtx v2.1f	
2018-03-08 ltcnts.dtx v1.1k		\document@default@language: Add to latexrelease (github/68) . . . . .	981
\@ifbothcounters: Interface added	400	2018-09-02 ltsect.dtx v1.1b	
\@removefromreset: Interface added	400	\@dottedtocline: Prevent protrusion ( <a href="https://tex.stackexchange.com/q/172785/10109">https://tex.stackexchange.com/q/172785/10109</a> ) . . . . .	754
\counterwithin: Interface added . . . . .	401	2018-09-24 fontdef.dtx v3.0b	
2018-03-24 ltclass.dtx v1.2e		General: Start LR-mode if necessary (git/49) . . . . .	561
\pkgcls@use@this@release: Use full file name for old release . . . . .	838	2018-09-24 ltmath.dtx v1.2b	
2018-03-25 ltfinal.dtx v2.1a		\smash: Start LR-mode if necessary (git/49) . . . . .	638
General: default to UTF-8 . . . . .	982	\phantom: Start LR-mode if necessary (git/49) . . . . .	638
\UseRawInputEncoding: Macro added	983	2018-09-24 ltspace.dtx v1.3h	
2018-03-27 ltclass.dtx v1.2f		\enspace: Start LR-mode if necessary (git/49) . . . . .	327
\endfilecontents: Use full file name for old release . . . . .	827	\leavevmode@ifvmode: Macro added (git/49) . . . . .	327
2018-04-06 ltfinal.dtx v2.1b		2018-09-26 ltdefns.dtx v1.5e	
\UseRawInputEncoding: Undo changes to \DeclareFontEncoding@ and definition of		\renew@command: Always explicitly generate a space after the csname and not rely on \noexpand to save tokens (git/41) . . . . .	84
\DeclareUnicodeCharacter . . . . .	983	2018-09-26 ltmiscen.dtx v1.1n	
2018-04-07 ltfinal.dtx v2.1c		\writefile: Sometimes mask the endline char when writing to files (github/73) . . . . .	620
\UseRawInputEncoding: Undefine \inputencodingname . . . . .	983	\add@percent@to@temptokena: Sometimes mask the endline char when writing to files (github/73) . . . . .	619
2018-04-08 ltclass.dtx v1.2g		\protected@file@percent: Sometimes mask the endline char when writing to files (github/73) . . . . .	619
\@ifl@ter: Strip leading spaces from dates. . . . .	801	2018-09-26 ltsect.dtx v1.1c	
2018-04-08 ltclass.dtx v1.2h		\addcontentsline: Sometimes mask the endline char when writing to files (github/73) . . . . .	752
\@onefilewithoptions: Pass expanded date . . . . .	834	2018-10-10 ltspace.dtx v1.3i	
2018-04-08 ltfinal.dtx v2.1d		\esphack: Don't introduce breakpoints if @nobreak is true	
General: Delay full UTF-8 handling to \everyjob . . . . .	984		
2018-04-11 ltcnts.dtx v1.1l			
\counterwithin: Correct default (issue/38) . . . . .	401		
2018-05-02 ltluatex.dtx v1.1g			
General: find_sfd_file removed . . . . .	62		
finish_synctex_callback added . . . . .	63		
glyph_not_found added . . . . .	63		
read_sfd_file removed . . . . .	62		
2018-05-08 ltclass.dtx v1.2i			
\pkgcls@parse@date@arg: Make suspicious rollback a warning not error: github issue 43 . . . . .	836		

	and after sections . . . . .	318	2019-06-18 lltuatem.dtx v1.1j
2018-10-11 ltmiscen.dtx v1.1o			General: finish_syntex_callback renamed finish_syntex . . . . . 63
\@@sverb: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69) . . . . .	632	font_descriptor_objnum_provider added . . . . . 63	
\@setupverbvisiblespace: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69) . . . . .	631	make_extensible added . . . . . 63	
\@verbvisiblespacebox: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69) . . . . .	631	new_graf added . . . . . 62	
\asciispace: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69) . . . . .	631	page_objnum_provider added . . . . . 63	
\verbatim*: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69) . . . . .	632	process_pdf_image_content added . . . . . 63	
\verbvisiblespace: Provide \verbvisiblespace such that it is usable in normal text (github/70) . . . . .	631	wrapup_run added . . . . . 63	
2018-10-21 lltuatem.dtx v1.1i			2019-07-01 ltclass.dtx v1.3a
\new_luafunction: Function added . . .	61	\endfilecontents: Support UTF8 and spaces in filecontents environment file name . . . . . 824	
2018-11-09 ltbibl.dtx LaTeX2e			2019-07-01 ltfiles.dtx v1.2a
\bibliography: Zap spaces in the argument as BibTeX doesn't support them (github/88) . . . . .	784	\IfFileExists: Support UTF-8 . . . . . 343	
2018-11-18 ltoutenc.dtx v2.0k			\includeonly: Support UTF-8 . . . . . 338
General: Provide \Hwithstroke and \hwithstroke . . . . .	388	\set@curr@file: Support UTF-8 . . . . . 342	
2018-11-19 ltoutenc.dtx v2.0k			2019-07-09 ltfssbas.dtx v3.2c
General: Added \Hwithstroke and \hwithstroke . . . . .	370	\DeclareErrorFont: Don't set any \@f@... macros . . . . . 424	
2018-11-28 ltoutput.dtx v1.4d			2019-07-09 ltfssini.dtx v3.1c
\@combinedblfloats: Unbox \@outputbox to preserve boxing level (github/94) . . . . .	931	General: Explicitly set some defaults . . . . . 540	
2018-12-30 lttab.dtx v1.1p			2019-08-22 ltxref.dtx v1.1l
\@tabclassz: Add extra \hskip to guard against an \unskip at the start of a c-column cell (gh/102)	702	\labelformat: Commanded moved from variorref.sty . . . . . 612	
2019-02-07 ltfilehook.dtx v1.1o			\Ref: Commanded moved from variorref.sty . . . . . 612
\unqu@tefilef@und: Expand \@filef@und before executing second argument (github/109) . . .	858	\refstepcounter: Allow \p@... to have an argument . . . . . 612	
2019-02-07 ltfiles.dtx v1.1o			2019-08-27 fontdef.dtx v3.0c
\@swaptwoargs: Helper macro added . . .	346	General: Various commands made robust throughout the file . . . . . 554	
2019-02-07 ltfssbas.dtx v3.2b			2019-08-27 ltboxes.dtx v1.3b
\define@newfont: Changed wording of warning (github/107) . . . . .	421	General: Various commands made robust . . . . . 666	
			\endfilecontents: Make various commands robust . . . . . 824
			2019-08-27 ltdefns.dtx v1.5f
			General: Make various commands robust . . . . . 106
			\MakeRobust: Make the assignments global as we may need to apply them inside a group . . . . . 90
			2019-08-27 ltfilehook.dtx v1.2b
			\unqu@tefilef@und: Make command robust . . . . . 858
			2019-08-27 ltfiles.dtx v1.2b
			\IfFileExists: Make command robust . . . . . 343
			2019-08-27 ltfssbas.dtx v3.2d
			General: Make various commands robust . . . . . 407

2019-08-27 ltfsdcl.dtx v3.0s	\DeclareMathAccent: Make math accents robust . . . . .	503	command already robust . . . . .	505
	\set@mathdelimter: Make math delimiters robust . . . . .	510	\endfilecontents: Support optional argument for filecontents . . . . .	824
2019-08-27 ltfsini.dtx v3.1d	General: Make various commands robust . . . . .	515	2019-09-14 ltfinal.dtx v2.1h \@ucclist: Expand UTF8 chars when case changing (github/177)	988
2019-08-27 ltidxglo.dtx v1.1f	General: Make \index and \glossary robust . . . . .	779	2019-09-16 ltxref.dtx v1.1m General: Correctly revert the \p@... change . . . . .	612
2019-08-27 ltlength.dtx v1.1d	General: Make various command robust . . . . .	405	2019-09-21 fontdef.dtx v3.0d General: Distangle alias (gh/184)	556, 560
2019-08-27 ltlogos.dtx v1.1j	\TeX: Make \TeX command robust .	329	2019-10-02 ltexpl.dtx v0.0 General: Initial version . . . . .	72
2019-08-27 ltmath.dtx v1.2c	General: Make various commands robust . . . . .	635	2019-10-02 ltfinal.dtx v2.2 General: Load ltexpl . . . . .	989
2019-08-27 ltmiscen.dtx v1.1p	General: Make various commands robust . . . . .	614	2019-10-02 ltluatex.dtx v1.1k General: linebreak_ filter is exclusive	62
	\begin: Make command robust . .	621	mlist_ to_ hlist is exclusive . . .	62
	\end: Make command robust . .	624	process_ rule is exclusive . . . . .	62
2019-08-27 ltoutput.dtx v1.4e	\@begindvibox: Make \AtBeginDvi robust . . . . .	910	2019-10-07 ltab.dtx v1.1q \extracolsep: This needs to expand	694
2019-08-27 ltpage.dtx v1.0l	General: Make various commands robust . . . . .	787	2019-10-11 ltfsstrc.dtx v3.0l \@font@aliasinfo: Added 'alias' size function . . . . .	478
2019-08-27 ltpictur.dtx v1.1m	General: Make various commands robust . . . . .	707	2019-10-18 ltclass.dtx v1.3d \load@onefilewithoptions: Initialize \...-h@0k only when loading the package or class (gh/198) . . . . .	816
	Remove several unnecessary \gdef definitions . . . . .	707	2019-10-22 ltluatex.dtx v1.1j General: page_objnum_provider and process_pdf_image_content classified data . . . . .	63
2019-08-27 ltsect.dtx v1.1d	General: Make various commands robust . . . . .	743	2019-10-25 ltmiscen.dtx v1.1q \add@percent@to@temptokena: Allow unbalanced conditionals in #1 (gh/202) . . . . .	619
2019-08-27 ltspace.dtx v1.3j	General: Make various commands robust . . . . .	310	2019-10-26 ltfiles.dtx v1.2d \iffileonpath: quote on openin . .	345
	Remove several unnecessary \gdef definitions . . . . .	682	\IfExists: don't quote name . .	343
2019-08-27 ltab.dtx v1.1q	General: Make various commands robust . . . . .	682	\IfExists@: quote on openin . .	344
	Remove several unnecessary \gdef definitions . . . . .	682	\set@curr@file: remove quotes . .	342
2019-08-30 lterror.dtx v1.2q	\conditionally@traceoff: Macro added . . . . .	289	2019-11-01 ltdirchk.dtx v1.3a \filename@parse: take last . not first	12
	\conditionally@traceon: Macro added . . . . .	289	2019-11-02 ltmiscen.dtx v1.1s \centercr: Make \centercr robust (gh/203) . . . . .	626
2019-09-09 ltfsdcl.dtx v3.0s	\DeclareMathSymbol: Allow definition if the math symbol was a		2019-11-02 ltspace.dtx v1.3k \normalcr: Make also \normalcr robust . . . . .	314

2019-11-09 ltfiles.dtx v1.2e \set@curr@file: expand and \string before removing quotes . . . . .	342	2020-01-22 lttextcomp.dtx v1.0b \tc@subst: The overall default is \textcompsubstdefault not \substdefault . . . . .	579
2019-11-10 ltmiscen.dtx v1.1r \add@percent@to@temptokena: fix to special comment catcodes (gh/202) . . . . .	619	2020-01-25 fontdef.dtx v3.0f General: Load t1enc.def last (gh/255)	545
2019-11-11 ltfiles.dtx v1.2f \@iffileonpath: make \@filef@und match quoting used on \openin .	345	2020-01-25 ltoutenc.dtx v2.0m General: Load each encoding file only once (gh/255) . . . . .	395
2019-11-22 ltoutenc.dtx v2.0l General: Avoid spurious if fontenc selects LY1 as default encoding (gh/199) . . . . .	396	2020-01-27 ltclass.dtx v1.3g \endfilecontents: Fix typo in error message . . . . .	826
2019-11-29 ltclass.dtx v1.3e \opr@videopackage: Protect package info text (gh/52) . . . . .	804	2020-01-28 ltclass.dtx v1.3h \endfilecontents: Allow spaces in option string and display only unknown options not the whole option list (gh/256) . . . . .	825
2019-12-17 fontdef.dtx v3.0e \mddefault: Set \bfdefault to "b" \shapedefault: Set \shapedefault explicitly to "n" . . . . .	548 549	2020-01-31 ltvers.dtx v1.1e General: Allow for upcoming format as pre-release 0 . . . . .	40
2019-12-17 ltntcmd.dtx v3.4c \textssc: Macro added . . . . .	570	2020-02-02 ltluatex.dtx v1.1l General: Add reverselist callback type glyph_info added . . . . .	65 63
2019-12-17 ltfsbas.dtx v3.2e \usefont: Don't call \fontseries or \fontshape . . . . .	416	page_order_index added . . . . . post_linebreak_filter is reverselist . . . . .	63 62
2019-12-17 ltfsini.dtx v3.1e General: Provide custom series settings a la mweights . . . . .	516	create_callback: Provide proper fallbacks for user-defined callbacks without user-provided default handler . . . . .	66
\DeclareEmphSequence: Provide \emph sequences . . . . .	534	\DeclareFontSeriesDefault: Clarified error text . . . . .	517
2019-12-18 ltoutenc.dtx v2.0m General: Don't fake \textcompwordmark; take default from T1 instead . . . . .	365	Corrected misspelled csname (gh/264) . . . . .	517
\add@accent: Avoid code that breaks \accent . . . . .	358	2020-02-05 ltfsini.dtx v3.1g General: Changed the package default to info (gh/262) . . . . .	596
2019-12-21 fontdef.dtx v3.0e General: Distangle alias (gh/184)	554–557	Ensure we are on a new format (gh/260) . . . . .	596
2020-01-05 ltclass.dtx v1.3f \endfilecontents: Support more write streams in LuaTeX gh/238	824	2020-02-07 ltfsini.dtx v3.1h \symbol: XeTeX-specific version to avoid bug in maths mode. . . . .	537
2020-01-11 ltfsini.dtx v3.1f \rmfamily: Streamlined implementation with hook . . . . .	529	2020-02-10 ltfsaxes.dtx v1.0c \fontseries: Switch \if@forced@series added . . . . .	443
\ttfamily: Streamlined implementation with hook . . . . .	529	\fontseriesforce: Switch \if@forced@series added . . . . .	443
2020-01-20 ltfsdcl.dtx v3.0t \set@mathdelimiter: fix for gh/251	510	\if@forced@series: Switch \if@forced@series added . . . . .	443
2020-01-20 ltoutenc.dtx v2.0n General: fix for gh/251 . . . . .	366	2020-02-10 ltfsini.dtx v3.1h \@defaultfamilyhook: Add \@defaultfamilyhook to \normalfont (gh/269) . . . . .	530

\reset@font: Add \@defaultfamilyhook to \normalfont (gh/269) . . . . .	538	\update@series@target@value: Drop surplus “m” from \reserved@d (gh/291) . . . . .	521, 522
2020-02-10 lttextcomp.dtx v1.0c General: Use \tabacckludge for tabbing where necessary (gh/271) 582		2020-02-27 ltdefns.dtx v1.5g \@gobblethree: Macro added . . . . .	87
2020-02-11 fontdef.dtx v3.0g General: Provide value for \@fontenc@load@list (gh/273) . 545		2020-02-27 ltfssaxes.dtx v1.0d \series@maybe@drop@one@m: Drop	
2020-02-11 ltfsini.dtx v3.1h General: Provide default value for \@fontenc@load@list (gh/273) . 540		“m” in certain values from a fixed list (gh/293) . . . . .	447
2020-02-11 ltoutenc.dtx v2.0o General: Update \@fontenc@load@list with option list (gh/273) . . . . .	396	\set@target@series: Drop “m” only in a specific set of values (gh/293)	446
2020-02-14 ltpictur.dtx v1.1n \linethickness: Suppress spaces following the declaration (gh/274) 712		2020-02-27 ltfssbas.dtx v3.2g \DeclareFontShape@: Only “m” if the series value is a member of a fixed list and issue warning if doing it (gh/293) . . . . .	408
2020-02-18 ltfsini.dtx v3.1i \bfseries: Make the \ifx selection outside of \fontseries argument so that it is not done several times 524		2020-03-02 ltxpl.dtx v1.0a General: Don’t load expl3 if already in the format (gh/295) . . . . .	72
\mdseries: Make the \ifx selection outside of \fontseries argument so that it is not done several times 525		2020-03-05 ltxpl.dtx v1.1 General: Load xparseltx if	
\prepare@family@series@update: No series auto-update when forced (gh/277) . . . . .	520	\NewDocumentCommand is not defined by expl3.ltx . . . . .	72
Recognize current family if it is not a “meta” family and auto-update series using \bfdefault (gh/277) 520		2020-03-06 ltboxes.dtx v1.3c \clap: Macro \clap added . . . . .	681
2020-02-18 ltmath.dtx v1.2d \mathindent: Make \mathindent a skip register to match amsmath (gh/252) . . . . .	647	2020-03-07 lthuatax.dtx v1.1m remove_from_callback: Do not call callback.register for user-defined callbacks . . . . .	68
\equation: Separate formula and eqn number by at least a space in fleqn option . . . . .	648	2020-03-07 ltmath.dtx v1.2e \negthinspace: Add amsmath math/text spacing commands to the kernel (gh/303) . . . . .	640
2020-02-20 ltclass.dtx v1.3j \endfilecontents: Fix missing quotes around file name (gh/284) . . . . .	826	2020-03-07 ltspace.dtx v1.3l General: Moved \thinspace, \negthinspace and \, to ltmath.dtx (gh/303) . . . . .	310
2020-02-24 ltfssbas.dtx v3.2f \DeclareFontShape@: Drop surplus “m” in series when defining fontshape (gh/289) . . . . .	408	2020-03-19 fontdef.dtx v3.0h General: Support legacy use of \bfdefault and \mddefault (gh/306) . . . . .	548
2020-02-25 ltfsini.dtx v3.1j \bf@def@ult: Drop surplus “m” from \bfdef@ult and \mddef@ult (gh/291) . . . . .	527	2020-03-19 ltfssdcl.dtx v3.0u \document@select@group: fix for (gnats/3357) . . . . .	491
\prepare@family@series@update: Drop surplus “m” from \target@series@value (gh/291) 521		2020-03-19 ltfsini.dtx v3.1k \DeclareFontSeriesDefault: Support legacy use of \bfdefault and \mddefault (gh/306) . . . . .	517
		\maybe@update@bfseries@defaults: Support legacy use of \bfdefault and \mddefault (gh/306) . . . . .	524
		\mdseries: Support legacy use of \bfdefault and \mddefault (gh/306) . . . . .	525

2020-04-06 ltfssini.dtx v3.1m	\@yargarraycr: Support calc syntax (gh/152) . . . . .	696
\bf@def@ult: Hook added (gh/306)	527	
\maybe@update@bfseries@defaults:		
Hook added (gh/306) . . . . .	524	
\maybe@update@mdseries@defaults:		
Hook added (gh/306) . . . . .	525	
2020-04-07 ltclass.dtx v1.3k		
\IfFormatAtLeastTF: Macro added; also in rollback (gh/168) . . . . .	800	
\load@onefile@withoptions: Use different method to ignore unprocessed options (gh/22) . . . . .	820	
\ProcessOptions*: Use different method to ignore unprocessed options (gh/22) . . . . .	809	
\RequirePackageWithOptions: Use different method to ignore unprocessed options (gh/22) . . . . .	812	
2020-04-09 ltfloat.dtx v1.2d		
\@textsubscript: Set non-zero baseline (gh/249) . . . . .	775	
\textsubscript: Set non-zero baseline (gh/249) . . . . .	774	
2020-04-13 ltfssdcl.dtx v3.0v		
\process@table: Small update for speed. . . . .	495	
2020-04-13 ltfssini.dtx v3.1n		
\init@series@setup: Handling \seriesdefault changes (gh/315)	523	
\seriesdefault@kernel: Handling \seriesdefault changes (gh/315)	541	
2020-04-21 ltmath.dtx v1.2f		
\@yeqnacr: Support calc syntax (gh/152) . . . . .	645	
2020-04-21 ltmiscen.dtx v1.1t		
\@icentercr: Support calc syntax (gh/152) . . . . .	627	
2020-04-21 ltpictur.dtx v1.1o		
\@istackcr: Support calc syntax (gh/152) . . . . .	712	
2020-04-21 ltspace.dtx v1.3m		
\@hspace: Support calc syntax (gh/152) . . . . .	327	
\@newline: Support calc syntax (gh/152) . . . . .	315	
\@vspace@calcify: Support calc syntax (gh/152) . . . . .	315	
\@vspacer: Support calc syntax (gh/152) . . . . .	324	
\addvspace: Support calc syntax (gh/152) . . . . .	321	
2020-04-21 lttab.dtx v1.1r		
\@itabcr: Support calc syntax (gh/152) . . . . .	688	
	\@raggedleft: Added \finalhyphendemerits setting (gh/247) . . . . .	627
	\centering: Added \finalhyphendemerits setting (gh/247) . . . . .	628
	\dischyp: Do not overwrite \-\ under LuaTeX . . . . .	105
	\typeout: Allow \par in the argument (gh/335) . . . . .	78
	\series@maybe@drop@one@cm: Need to use \edef (gh/336) . . . . .	447
	\IfFontSeriesContextTF: Macros added (gh/335) . . . . .	532
	\centering: Added \finalhyphendemerits setting (gh/247) . . . . .	627

\raggedright: Added \finalhyphendemerits setting (gh/247) . . . . .	627	2020-08-02 lltuaretex.dtx v1.1q \newattribute: Move reset to 0 inside conditional . . . . .	50
2020-06-04 ltexpl.dtx v1.2c General: Define a local version of some L <sup>A</sup> T <sub>E</sub> X 2 <sub><math>\varepsilon</math></sub> basic macros to support package loading . . . . .	72	\newluabytocode: Move reset to 0 inside conditional . . . . .	53
2020-06-04 ltfinal.dtx v2.2a General: Load ltexpl in ltdefns . . .	989	\newluachunkname: Move reset to 0 inside conditional . . . . .	54
2020-06-05 ltclass.dtx v1.3l \@currnamestack: Added \@expl@pop@filename@@ . . . . .	798	\newluafunction: Move reset to 0 inside conditional . . . . .	52
Added \@expl@push@filename@@ and \@expl@push@filename@aux@@ . . .	797	\newwhatsit: Move reset to 0 inside conditional . . . . .	53
2020-06-05 ltfiles.dtx v1.2h \document: Added hook to load l3backend code . . . . .	332	2020-08-08 ltclass.dtx v1.3m \endfilecontents: define \q@curr@file directly as the quotes have already been removed (gh/220) . . . . .	825
2020-06-10 lltuaretex.dtx v1.1n General: Define \@gobble/\@firstofone even for L <sup>A</sup> T <sub>E</sub> X to allow early loading. . .	48	2020-08-10 lltuaretex.dtx v1.1r General: Load lltuaretex Lua module during format building . . . . .	54
2020-07-04 ltoutenc.dtx v2.0q General: Implement \remove@tlig in LuaTeXwithout font reloading . .	383	2020-08-15 ltpictur.dtx v1.2a \@defaultunitsset: Macro added . .	709
2020-07-08 ltexpl.dtx v1.2d General: Add a last-minute hook for expl3 . . . . .	73	2020-08-19 ltdefns.dtx v1.5k \@carcube: Made \long for \NewCommandCopy . . . . .	81
2020-07-08 ltfinal.dtx v2.2b General: Add a last-minute hook for expl3 . . . . .	980	\robust@command@act: Made \robust@command@act (was \declare@command@copy) more generic . . . . .	94
2020-07-27 ltmath.dtx v1.2g \cases: Don't make the command \long (gh/354) . . . . .	639	\ShowCommand: Added \ShowCommand (gh/373) . . . . .	97
\matrix: Don't make the command \long (gh/354) . . . . .	639	2020-08-19 ltexpl.dtx v1.2e General: Add	
\pmatrix: Don't make the command \long (gh/354) . . . . .	639	\@expl@cs@{thing}@spec@@N for \ShowCommand (gh/373) . . . . .	76
2020-07-27 ltoutenc.dtx v2.0r \use@text@encoding: Don't make the command \long (gh/354) . .	360, 361	Add \@expl@cs@to@str@@N and \@expl@str@if@eq@@nnTF for \NewCommandCopy (gh/239) . . .	75
2020-07-27 ltpage.dtx v1.0m \markright: Don't make the command \long (gh/354) . . . . .	788	2020-08-20 lplain.dtx v2.3d \alloc@: Define \alloc@ in terms of \@alloc . . . . .	22
2020-07-27 ltpictur.dtx v1.1p \linethickness: Don't make the command \long (gh/354) . . . .	712	2020-08-21 ltclass.dtx v1.3o General: Integration of new hook management interface . . . . .	791
2020-07-27 ltsect.dtx v1.1e \author: Don't make the command \long (gh/354) . . . . .	743	2020-08-21 ltdefns.dtx v1.5l General: Integration of new hook management interface . . . . .	78
\date: Don't make the command \long (gh/354) . . . . .	743	2020-08-21 ltdefns.dtx v1.5m \MakeRobust: Make \MakeRobust produce the same command structure as \DeclareRobustCommand . . . . .	90
2020-08-01 lltuaretex.dtx v1.1p General: new_ graf is exclusive . .	62	2020-08-21 ltexpl.dtx v1.2d General: Dropped unused command .	72

2020-08-21 ltfiles.dtx v1.2i		2020-09-30 ltfssini.dtx v3.2d	
General: Integration of new hook management interface . . . . .	330	\maybe@update@bfseries@defaults: \bfdefault@previous not \bfseries@previous (gh/395) . . .	524
2020-08-21 ltfinal.dtx v2.2i		\mdseries: \mddefault@previous not \mdseries@previous (gh/395) . . .	525
General: Integration of new hook management interface . . . . .	974	2020-10-01 ltclass.dtx v1.3r	
2020-08-21 ltfssaxes.dtx v1.0g		\@pr@videopackage: Allow for package substitution . . . . .	804
General: Integration of new hook management interface . . . . .	454	2020-10-01 ltsect.dtx v1.1e	
2020-08-21 ltfssini.dtx v3.2b		\addcontentsline: add a fourth argument for better hyperref compatibility . . . . .	752
\bf@def@ult: Integration of new hook management interface . . . . .	527	2020-10-04 ltfiles.dtx v1.2j	
\mdseries@defaults: Integration of new hook management interface	530	\@include: Quotes around the aux file name removed, they are not needed and upset BibTeX (gh/400) . . . . .	340
\maybe@update@bfseries@defaults: Integration of new hook management interface . . . . .	524	2020-10-04 lthooks.dtx v1.0d	
\maybe@update@mdseries@defaults: Integration of new hook management interface . . . . .	525	General: Definition \AddToHookNext was supposed to be for \AddToHook vice versa (gh/401) . . . . .	253
\reset@font: Integration of new hook management interface . . . . .	538	2020-10-08 ltclass.dtx v1.3s	
\rmfamily: Integration of new hook management interface . . . . .	529	\@currnamestack: Added missing 2020/02/02 \IncludeInRelease . . .	797
\ttfamily: Integration of new hook management interface . . . . .	529	2020-10-11 ltclass.dtx v1.3t	
2020-08-21 ltmiscen.dtx v1.1v		\load@onefilewithoptions: Restore \currpkg@reqd after finished loading a package file (gh/408). . .	818
General: Integration of new hook management interface . . . . .	614	2020-10-18 ltclass.dtx v1.3t	
2020-08-21 ltoutput.dtx v1.4f		\PassOptionsToClass: Drop path from \input@path (gh/414). . . .	805
\@begindivbox: Integration of new hook management interface . . .	910	2020-10-23 ltmiscen.dtx v1.1w	
2020-08-23 ltxref.dtx v1.1o		\enddocument: Make enddocument/afteraux one-time	616
\refstepcounter: add default definition of \@currentcounter .	612	2020-10-26 ltmiscen.dtx v1.1x	
2020-09-06 ltclass.dtx v1.3q		\@kernel@before@enddocument: \enddocument should always start out in vmode (gh/385) . . . . .	618
\load@onefilewithoptions: Save \currpkg@reqd so that we don't lose track of package substitutions. . . . .	817	2020-11-09 ltclass.dtx v1.3u	
2020-09-06 ltdefns.dtx v1.5n		\pkgcls@rollbackdate@error: Change help text because the package may have existed then — there is just no rollback data (gh/423). . . . .	839
\char@if@alph: Macro added . . . .	104	2020-11-09 ltmath.dtx v1.2h	
2020-09-06 ltexpl.dtx v1.2f		\@expl@char@generate@cn \negthickspace, \negmedspace and \negthickspace have been only in amsmath, so we need to undefine for rollback (gh/423) . . . . .	641
General: Add \@expl@str@map@function@nn and \@expl@ for \string@makeletter (gh/386)	76	2020-11-20 ltclass.dtx v1.3u	
2020-09-09 ltshipout.dtx v1.0b		\@currpath: Macro added . . . . .	796
\_\_shipout\_picture\_overlay:n: Prevent overfull box warnings (gh/387) . . . . .	890	\@kernel@currpathstack: Macro added . . . . .	799
2020-09-26 ltfinal.dtx v2.2j			
General: Load first aid file if existing	991		

\load@onefile@withoptions: Copy option list to the requested package. . . . .	819	\fontseries: Distangle series and shape update (gh/444) . . . . .	443
\PassOptionsToClass: Copy option list to the requested package. . . . .	806	\fontshape: Distangle series and shape update (gh/444) . . . . .	451
\ProvidesPackage: Use string comparison instead of \ifx . . . . .	803	\fontshapeforce: Distangle series and shape update (gh/444) . . . . .	451
2020-11-20 ltcmd.dtx v1.0a General: Initial version derived from xparse.dtx . . . . .	108	2020-12-04 ltfsini.dtx v3.2f General: Adjust start values for series and shape (gh/444) . . . . .	540
2020-11-20 ltfilehook.dtx v1.0d \unqu@tefilef@und: Move loading to @input@file@exists@with@hooks and expand \@filef@und to avoid getting the wrong file name in the case of a substitution. . . . .	857	2020-12-10 ltbibl.dtx v1.1s \nocite: Delay any \nocite in the preamble instead of raising an error . . . . .	785
2020-11-23 ltshipout.dtx v1.0d \_shipout_execute_cont:: Check for both kernel and user hook (gh/431) . . . . .	880	2020-12-10 ltfsbas.dtx v3.2h \usefont: Drop “m” if the series value is a member of a fixed list and issue warning if doing it (gh/453) . . . . .	416
\_shipout_execute_main_cont:Nnnn: Check for both kernel and user hook (gh/431) . . . . .	882	2020-12-14 ltclass.dtx v1.3v \currnamestack: Removed @expl@@hook@curr@name@push@on . . . . .	797
2020-11-24 ltexpl.dtx v1.2g General: Support for roll forward (gh/434) . . . . .	74–76	2020-12-18 ltexpl.dtx v1.2h \@kernel@after@enddocument@afterlastpage: Define kernel \enddocument hooks early . . . . .	72
2020-11-24 ltfilehook.dtx v1.0d General: Support for roll forward (gh/434) . . . . .	855	2020-12-22 ltfsaxes.dtx v1.0h \delayed@merge@font@series: Distangle series and shape update (gh/444) . . . . .	445, 446
2020-11-24 lthooks.dtx v1.0f \_hook_end_document_label_check:: Support for roll forward (gh/434) . . . . .	215	\delayed@merge@font@shape: Distangle series and shape update (gh/444) . . . . .	452
2020-11-24 ltshipout.dtx v1.0d General: Support for roll forward (gh/434) . . . . .	894	2020-12-22 lfsstrc.dtx v3.0n \selectfont: Execute delayed series and shape updates (gh/444) . . . . .	459
\AtBeginDvi: Support for roll forward (gh/434) . . . . .	893	2021-01-07 ltfilehook.dtx v1.0e General: Added rollback for this case to avoid spurious errors (part of gh/463) . . . . .	868
2020-11-25 ltdefns.dtx v1.5o \carcube: Added missing latexrelease entry . . . . .	81	\unqu@tefilef@und: Restore \CurrentFile(Path)(Used) after the input (gh/464) . . . . .	858
2020-12-02 ltluatex.dtx v1.1s General: Fix return value of list callbacks . . . . .	65	2021-01-07 lthooks.dtx v1.0h \_hook_strip_double_slash:w: Assume hook name has at least three nonempty parts (gh/464) . . . . .	223
2020-12-03 lfsstrc.dtx v3.0m \selectfont: Install a hook in \selectfont (gh/444) . . . . .	460	\_hook_tl_set:cx: Manually define some l3tl commands to work around expl3 changes . . . . .	206
2020-12-04 ltfilehook.dtx v1.0d \undeclare@file@substitution: Don’t drop file substitution commands on rollback . . . . .	860	2021-01-08 ltshipout.dtx v1.0f \_shipout_execute_cont:: Added another kernel hook for more flexibility (cf	
2020-12-04 ltfsaxes.dtx v1.0h General: Reorganized the rollback data . . . . .	434		

<a href="https://github.com/pgf-tikz/pgf/issu#1960">https://github.com/pgf-tikz/pgf/issu#1960</a>	15	ltfssdcl.dtx v3.0w
.....	880	\document@select@group: fix for (gh/501) .....
2021-01-10 ltshipout.dtx v1.0g		491
\@kernel@after@shipout@background: Internal hook		
\@kernel@after@shipout@background added .....	884	
\RawShipout: Macro added .....	884	
2021-01-19 ltshipout.dtx v1.0h		
\__shipout_run_firstpage_hook:: Handling of firstpage hook altered	884	
2021-01-21 ltclass.dtx v1.3w		
\@kernel@currpathstack: Add empty entry for latexrelease .....	799	
2021-01-21 ltexpl.dtx v1.3a		
General: Move <code>xparse</code> rollback code to <code>ltcmd.dtx</code> .....	75	
2021-01-21 ltfinal.dtx v2.2l		
General: Load <code>glyptounicode.tex</code> for <code>pdfTeX</code> .....	981	
2021-01-22 ltshipout.dtx v1.0i		
\__shipout_finalize_box:: Add <code>pre_shipout_filter</code> Lua callback .....	879	
2021-01-24 ltexpl.dtx v1.3a		
General: Define <code>expl3</code> hooks conditionally .....	72	
2021-01-31 ltfilehook.dtx v1.0f		
\@curr@file@reqd: set <code>\protect</code> to <code>\string gh/481</code> .....	861	
2021-02-03 ltfloat.dtx v1.2e		
\@savemarbox: Explicitly end with <code>\par</code> (gh/489) .....	769	
2021-02-04 ltboxes.dtx v1.4b		
\color@endbox: Always add the color groups (gh/488) .....	670	
2021-02-08 ltfilehook.dtx v1.0g		
\unqu@tefilef@nd: Undo the internal for robust <code>\InputIfFileExists</code> in rollback (gh/494) .....	859	
2021-02-08 ltmiscen.dtx v1.1y		
\end: Undo the internals for robust <code>\begin</code> and <code>\end</code> in rollback (gh/494) .....	625	
2021-02-10 ltboxes.dtx v1.4b		
\@mpfootnotetext: Explicitly run <code>\par</code> in support for paragraph tagging .....	678	
2021-02-10 ltfloat.dtx v1.2e		
\@footnotetext: Explicitly run <code>\par</code> at the end of footnote text in preparation for paragraph hooks	775	
		\document@select@group: fix for (gh/501) .....
		491
		\footref: \footref added .....
		778
		2021-02-17 ltoutenc.dtx v2.0t
		General: Adjust values for \textasteriskcentered To match TS1 definition (gh/502) .....
		388
		Special definition for \textasteriskcentered when missing in TS1 (gh/502) .....
		379
		2021-02-18 ltclass.dtx v1.3x
		\@fileswithoptions: save raw class option list (gh/85) .....
		814
		\@remove@eq@value: macro added (gh/85) .....
		807
		\@use@option: value from unused option list (gh/85) .....
		810
		\@optionNotUsed: value from unused option list (gh/85) .....
		807
		\@PassOptionsToClass: save raw option lists (gh/85) .....
		805
		2021-02-19 ltoutenc.dtx v2.0u
		General: Add \textnonbreakinghyphen, \textfiguredash and \texthorizontalbar (gh/404) .....
		368, 372, 387
		2021-02-25 ltfinal.dtx v2.2m
		General: Improve speed of ToUnicode everyjob loading code .....
		981
		2021-03-03 ltclass.dtx v1.3y
		\endfilecontents: Fix overwrite check for files with UTF-8 (gh/415) .....
		826
		2021-03-05 ltclass.dtx v1.3z
		\ProcessOptions*: modify so braces to not give errors (gh/513) .....
		808
		2021-03-12 ltfiles.dtx v1.2k
		\IfFileExists@: Allow unbalanced conditionals (gh/530) .....
		344
		2021-03-18 ltcmd.dtx v1.0b
		General: Use \NewModuleRelease. .
		108
		2021-03-18 ltfilehook.dtx v1.0h
		\__filehook_file_pop_assign:nnnn: Define \g_@@_input_file_seq to avoid losing data when rolling back. .....
		855
		2021-03-18 ltfssaxes.dtx v1.0i
		General: Fix rollforward definition. .
		444
		2021-03-18 ltfssini.dtx v3.2g
		General: Add legacy hook definitions for rollback. .....
		530

2021-03-18 lthooks.dtx v1.0i	\_hook\_end\_document\_label\_check:: Only add top-level if not already there. .... 215	2021-04-20 ltfilehook.dtx v1.0i	\@curr@file@reqd: Make expand to a string (tracks change in l3kernel) .... 861
	Remove the (empty) “top-level” from \@currnamestack. .... 215	2021-04-26 ltfssbas.dtx v3.2i	\usefont: Unconditionally switch to the requested font face (gh/444) 416
	General: Use \NewModuleRelease. . 204	2021-04-26 ltfssini.dtx v3.2h	\reset@font: Unconditionally switch to the requested font face (gh/444) .... 538
2021-03-18 ltvers.dtx v1.1f	\@check@IncludeInRelease: Add support for usage in \NewModuleRelease ..... 41	2021-04-26 ltfsstrc.dtx v3.0o	\selectfont: Unset the forced series boolean when reaching \selectfont (gh/444) .... 460
	\new@moduledate: Added \NewModuleRelease. .... 42	2021-04-29 lthooks.dtx v1.0m	\ActivateGenericHook: Add \ProvideHook etc. .... 249
2021-03-19 lttextcomp.dtx v1.0e	General: Use \NewModuleRelease .. 576	2021-04-29 ltoutenc.dtx v2.0v	General: Add composites for \ae/\AE/\æ/\鏗 (gh/552) .... 392
2021-03-26 ltplain.dtx v2.3e	\@unused: Allocate \@inputcheck and \@unused early so that they are before expl3 allocates more streams (gh/538) .... 24	2021-05-18 ltclass.dtx v1.4b	\@raw@classoptionslist: Initialise to \relax to match \@classoptionslist ..... 796
	2021-03-27 ltclass.dtx v1.4a	2021-05-24 ltcmd.dtx v1.0e	General: Use \msg_... instead of \_kernel_msg.... 108
	\@currnamestack: Do not completely roll back if expl3 is loaded. .... 798	2021-05-24 ltcmdhooks.dtx v1.0b	General: Use \msg_... instead of \_kernel_msg.... 257
2021-04-16 ltvers.dtx v1.1g	\new@moduledate: \NewModuleRelease with the same arguments as \IncludeInRelease. .... 42	2021-05-24 ltfilehook.dtx v1.0k	General: Use \msg_... instead of \_kernel_msg.... 853
2021-04-17 ltfiles.dtx v1.2m	\@kernel@after@begindocument: Move \@kernel@before@begindocument and \@kernel@after@begindocument init earlier so that other modules can write to the hooks ..... 335	2021-05-24 lthooks.dtx v1.0n	General: Use \msg_... instead of \_kernel_msg.... 204
	2021-04-18 ltluatex.dtx v1.1t	2021-05-24 ltpara.dtx v1.0g	General: Use \msg_... instead of \_kernel_msg.... 300
	General: input_level_string added .. 63	2021-05-26 ltdefns.dtx v1.5p	\MakeRobust: Normalize error message in \MakeRobust ..... 90
2021-04-18 ltplain.dtx v2.3f	\loggingall: 3 ..... 35	2021-06-03 ltclass.dtx v1.4c	\@kernel@currpathstack: Take care of \@kernel@currpathstack when rolling back/forward. .... 799
	Drop pre- $\varepsilon$ - $\text{\TeX}$ support ..... 35	\@kernel@currpathstack: Take care of \@kernel@currpathstack when rolling back/forward. .... 799	2021-06-04 ltcmd.dtx v1.0f
	\tracingnone: 3 ..... 36	General: Normalize various error messages ..... 171	General: Normalize error message ..... 632
	Drop pre- $\varepsilon$ - $\text{\TeX}$ support ..... 36	2021-06-05 ltmiscen.dtx v1.1z	
2021-04-19 ltcmd.dtx v1.0d	\_cmd_cmd_type_cases:Nnnnn: Renamed \_cmd_cmd_if_xparse:NTF to \_kernel_cmd_if_xparse:NTF for cross-module usage ..... 169	\@sverb: Normalize error message ..	
	2021-04-20 ltexpl.dtx v1.3c		
	\@kernel@after@enddocument@afterlastpage: Don't empty kernel hooks on rollback ..... 72		

2021-06-06 ltclass.dtx v1.4c	\@loadwithoptions: handle raw options for gh/580 . . . . .	812	2021-07-23 ltclass.dtx v1.4e	\load@onefile@withoptions: Make class/name/after a one-time hook . . . . .	820	
	\load@onefile@withoptions: Copy raw options for gh/580 . . . . .	819		Make class/name/before a one-time hook . . . . .	819	
	\PassOptionsToClass: apply \expandafter to raw options for gh/580 . . . . .	806		Make package/name/after a one-time hook . . . . .	820	
2021-06-09 ltclass.dtx v1.4b	\endfilecontents: Use \@latex@note@no@line to display the information . . . . .	825, 826		Make package/name/before a one-time hook . . . . .	819	
	\@latex@note@no@line: Macros added . . . . .	284	2021-07-23 lfiles.dtx v1.2n	\@include: Make include/name/after a one-time hook . . . . .	340	
2021-06-09 lterror.dtx v1.2r	\@latex@note@no@line: Improve information message . . . . .	408		Make include/name/before a one-time hook . . . . .	340	
	\counterwithin: New implementation for \counterwithout and \counterwithin . . . . .	400		Make include/name/end a one-time hook . . . . .	340	
2021-07-08 ltcnts.dtx v1.1m	\PackageNoteNoLine: Provide \ClassNote and \PackageNote . . . . .	283	2021-07-27 lthooks.dtx v1.0o	\_hook_clear_next:n: Macro made public . . . . .	241	
	\fileswithoptions: add \unexpanded . . . . .	815		\ClearHookNext: Macro added . . . . .	249	
2021-07-11 lterror.dtx v1.2s	\classoptionslist: Drop \@onlypreamble . . . . .	796	2021-07-28 ltsect.dtx v1.1f	\contentsline: Pick up four arguments (gh/633) . . . . .	753	
	\ifclasslater: Drop \@onlypreamble . . . . .	800	2021-07-30 lcmd.dtx v1.0d	\cmd_cmd_type_cases:Nnnnn: Added \@@_cmd_type_cases:Nnnnn for \NewCommandCopy and \ShowCommand support . . . . .	169	
2021-07-12 ltclass.dtx v1.4d	\ifclassloaded: Drop \@onlypreamble . . . . .	800		2021-07-31 ltoutput.dtx v1.4e	\f1@tracemessage: Enable display when doing \tracefloatvals . . . . .	957
	\ifclasswith: Drop \@onlypreamble . . . . .	802		2021-07-31 ltoutput.dtx v1.4g	\ShowFloat: Macro added . . . . .	955
	\ifl@ter: Drop \@onlypreamble . . . . .	801	2021-08-02 lthooks.dtx v1.0o	\ActivateGenericHook: Change name . . . . .	249	
	\pkgextension: Drop \@onlypreamble . . . . .	796		\DisableGenericHook: Change name . . . . .	249	
	\optionlist: Drop \@onlypreamble . . . . .	800	2021-08-07 lcmd.dtx v1.0g	\cmd_add_grabber:N: Replicate argument processors for all embellishments (gh/639) . . . . .	132	
	\unusedoptionlist: Drop \@onlypreamble . . . . .	796		\cmd_add_type_E:w: Replicate argument processors for all embellishments (gh/639) . . . . .	130	
	\IfFormatAtLeastTF: Drop \@onlypreamble . . . . .	800	2021-08-08 ltfinal.dtx v2.2p	\IfPDFManagementActiveTF: Default definition added (gh/640) . . . . .	990	
2021-07-20 lcmdhooks.dtx v1.0c	\_hook_patch_DeclareRobustCommand:Nnn: Use \robust@command@chk@safe before \if@newcommand. . . . .	262	2021-08-10 ltvers.dtx v1.1h	\check@IncludeInRelease: Add error to aid debugging . . . . .	41	
	\_hook_gremove_code:nn: Do not queue removals (gh/625) . . . . .	224		2021-08-11 lluatex.dtx v1.1u	General: Define missing local function . . . . .	56
2021-07-22 lthooks.dtx v1.0o	\_hook_hook_gput_code_do:nnn: Do not queue removals (gh/625) . . . . .	217				

2021-08-20 lterror.dtx v1.2t	\@badend: Improve \@badend error message (gh/587) . . . . .	287	2021-09-03 ltoutput.dtx v1.4h	\ShowFloat: Renamed, original name never distributed . . . . .	955
2021-08-20 lhooks.dtx v1.0p	General: Added deprecation warnings for old generic hook commands (gh/638) . . . . .	251	2021-09-06 ltfinal.dtx v2.2q	\@cuclclist: Correctly upper and lowercase \ij and \IJ (gh/658) .	988
	Documentation updates for generic hook commands (gh/638) . . . . .	182	2021-09-06 lhooks.dtx v1.0r	\__hook_hook_gput_code_do:n:nn: Use dedicated conditional (gh/606) .	217
	Renames of generic hook commands (gh/638) . . . . .	211		\__hook_if_execute_immediately:n: Macro added (gh/606) . . . . .	244
	Section on generic hooks added (gh/638) . . . . .	197		\__hook_use_once:n: Clean up after \UseOneTimeHook (gh/606) . . .	243
2021-08-25 ltclass.dtx v1.4f	General: Standardise generic hook names (gh/648) . . . . .	791		\__hook_use_once_clear:n: Clean up after \UseOneTimeHook (gh/606)	243
	\load@onefile@withoptions: Declare non-generic package and class hooks . . . . .	820	2021-09-10 lfssini.dtx v3.2i	\bfseries: Do delayed changes to \bfdefault in a separate macro for better reuse (gh/663) . . . . .	524
2021-08-25 lcmdhooks.dtx v1.0d	\__hook_try_put_cmd_hook:w: Simplify generic hook detection .	259		\DeclareFontSeriesDefault: Do delayed changes to \bfdefault or \mddefault first (gh/663) . . . . .	517
2021-08-25 lfilehook.dtx v1.0l	\unqu@tefilef@nd: Declare non-generic file hooks . . . . .	858	\maybe@update@bfseries@defaults:	Do delayed changes to \bfdefault in a separate macro for better reuse (gh/663) . . . . .	524
2021-08-25 lfiles.dtx v1.2o	\Qinclude: Declare non-generic include hooks . . . . .	341	\maybe@update@mdseries@defaults:	Do delayed changes to \mddefault in a separate macro for better reuse (gh/663) . . . . .	525
	Standardise generic hook names (gh/648) . . . . .	340	\mdseries: Do delayed changes to \mddefault in a separate macro for better reuse (gh/663) . . . . .	525	
2021-08-25 lhooks.dtx v1.0p	\__hook_try_declarng_generic_next_hook:nn: Standardise generic hook names (gh/648) . . . . .	219	2021-09-12 lftntcmd.dtx v3.5a	\check@nocorr@: use \unexpanded to make # safe . . . . .	571
2021-08-27 lcmd.dtx v1.0h	\NewDocumentEnvironment: Check for end-of-environment command .	177	2021-09-12 ltoutenc.dtx v2.0w	General: Move zero skip between i and j for hyphenation (gh/658) . . . . .	369
2021-08-27 lfilehook.dtx v1.0l	\__filehook_file_pop_assign:nnnn: Internal message name changes .	855	2021-09-18 ltpara.dtx v1.0i	\para_end:: Use skip rather than kern as guard. . . . .	304
	\__filehook_file_subst_cycle_error:cN: Use \msg... not \__kernel_msg... . . . . .	866	2021-09-26 lfssdcl.dtx v3.0x	\nfss_init_mv_freeze:N: Macro added for (gh/676) . . . . .	492
2021-08-27 lhooks.dtx v1.0q	General: Internal message name changes . . . . .	246		\c@localalphabets: Counter added for (gh/676) . . . . .	490
2021-08-27 ltpara.dtx v1.0i	General: Internal message name changes . . . . .	306	\document@select@group: Test if we should freeze the version (gh/676)	. . . . .	490
2021-08-30 lcmd.dtx v1.0h	General: Added support for \NewCommandCopy . . . . .	137	\freeze@math@version: Macro added for (gh/676) . . . . .	491	
	Added support for \ShowCommand . . . . .	142	2021-09-28 lcmdhooks.dtx v1.0e	\__hook_make_prefixes:w: Make patching of commands a global	

operation (gh/674) . . . . .	265	2021-11-30 ltexpl.dtx v1.3d
\_\_hook\_patch\_retokenize:Nnnn:		\skippeval: Moved over from xfp package (gh/711) . . . . . 76
Make patching of commands a global operation (gh/674) . . . . .	271	
2021-09-28 lthooks.dtx v1.0s		2021-12-02 ltcmd.dtx v1.0i
\_\_hook\_if\_usable\_use:n: Correct usage of older		\_\_cmd\_run\_code:: Correct defaults for optional arguments in end-of-environment code (gh/712) 116
\@@if_file_hook:wTF (gh/675) 243		\_\_cmd_start_aux:ccnnnn: Correct defaults for optional arguments in end-of-environment code (gh/712) 116
\_\_hook_try_declar ing_generic_hook_split:nNNnn: Correct usage of older		
\@@if_file_hook:wTF (gh/675) 219		
2021-10-14 ltboxes.dtx v1.4c		2021-12-07 ltexpl.dtx v1.3d
\@mpfootnotetext: Explicitly set \@currentcounter (gh/687) . . . . .	678	\skippeval: Added \dimeval and \skippeval (gh/711) . . . . . 77
2021-10-14 ltfiles.dtx v1.2p		2021-12-11 ltdirchk.dtx v1.3a
\@include: Warn about use in preamble . . . . .	340	General: Add comment lines into latex.ltx to indicate temp definitions that are later overwritten (gh/725) . . . . . 2
2021-10-14 ltfloat.dtx v1.2g		2021-12-12 ltoutenc.dtx v2.0y
\@footnotetext: Explicitly set \@currentcounter (gh/687) . . . . .	775	General: \DeclareUnicodeAccent now makes the encoding argument implicit (gh/253) . . . . . 385
2021-10-14 ltmath.dtx v1.2j		Added \DeclareUnicodeCommand and \DeclareUnicodeSymbol (gh/253) . . . . . 385
\@eqnse1: Explicitly set \@currentcounter (gh/687) . . . . .	645	
\@eqnarray: Explicitly set \@currentcounter (gh/687) . . . . .	649	
2021-10-15 ltfsdcl.dtx v3.0y		2021-12-27 ltluatex.dtx v1.1x
\DeclareMathVersion: Initialize variable for freezing math version (gh/676) . . . . .	495	\newprotectedluacmd: Macros added 53
2021-10-15 ltluatex.dtx v1.1v		2021-12-28 ltexpl.dtx v1.3d
General: provide_charproc_data added . . . . .	63	\ExpandArgs: Added document level names for \exp_args:Nc and the like (gh/735) . . . . . 77
2021-10-16 ltoutenc.dtx v2.0x		2021-13-31 ltcmd.dtx v1.0j
\add@accent: Dont set \spacefactor in math mode gh/643 . . . . .	358	\_\_cmd_show_environment:N: Make \ShowCommand stop for interaction . . . . . 142, 143
2021-10-19 ltpara.dtx v1.0k		2022-01-06 ltexpl.dtx v1.3e
General: Remove content from \tex_everypar:D on rollback . . . . .	307	\ExpandArgs: Adjust document level names for \exp_args:Nc and the like (gh/735) . . . . . 77
2021-10-20 ltcmdhooks.dtx v1.0f		2022-01-06 ltskipout.dtx v1.0l
\_\_hook_make_prefixes:w: Correct patching by expansion+redefinition when the macro contains a parameter tokens (gh/697). . . . .	263	\AtBeginShipoutNext: Correctly simulate \AtBeginShipout and \AtBeginShipoutNext without extending the syntax . . . . . 896
2021-11-17 ltluatex.dtx v1.1w		\AtEndDvi: Correctly simulate \AtEndDvi without extending the syntax . . . . . 895
General: hpack\_ quality is exclusive 62		2022-01-15 ltkeys.dtx v1.0b
Never pass on true return values for list callbacks . . . . .	65	\_\_keys_options:nN: Clear option list in end-of-package hook . . . . . 843
vpack\_ quality is exclusive . . . . .	62	2022-01-25 lplain.dtx v2.3h
2021-11-30 ltclass.dtx v1.5a		\obeyspaces: Provide redirection to support special use cases (gh/367) 29
\@onefilewithoptions@clashchk:		
Separated out from \@onefilewithoptions . . . . .	818	

2022-02-05 ltkeys.dtx v1.0b	friends . . . . .	179
.usage: Create properties in <code>ltlkeys</code>	842	
2022-02-07 ltkeys.dtx v1.0c		
.usage: Correct ..code property . . .	842	
2022-02-15 ltkeys.dtx v1.0c		
\DeclareKeys: Expand module argument . . . . .	846	
\DeclareUnknownKeyHandler: Added \DeclareUnknownKeysHandler . . .	846	
\ProcessKeyPackageOptions: Expand module argument . . . . .	846	
\SetKeys: Expand module argument . . . . .	847	
2022-02-16 ltkeys.dtx v1.0d		
\DeclareKeys: Allow for active characters in module argument . . .	846	
\DeclareUnknownKeyHandler: Allow for active characters in module argument . . . . .	846	
Rename \DeclareUnknownKeysHandler to \DeclareUnknownKeyHandler . . . . .	846	
\ProcessKeyPackageOptions: Allow for active characters in module argument . . . . .	846	
\SetKeys: Allow for active characters in module argument . . . . .	847	
2022-02-19 ltcmd.dtx v1.0k		
\IfBlankTF: Added \IfBlankTF and		
2022-02-20 ltfinal.dtx v2.2r		
\@uc1clist: Use \Expl@text@uppercase@@n, removing local redefinition of \UTF@two@octets@noexpand . . . . .	988	
use \text_lowercase:n . . . . .	987	
2022-02-21 ltkeys.dtx v1.0e		
\__keys_options_expand_module:Nn: Faster approach to module expansion . . . . .	845	
2022-02-28 ltcmd.dtx v1.0k		
General: Move latexrelease redefinitions from ltcmd.dtx . . . . .	180	
2022-02-28 ltexpl.dtx v1.3f		
General: Move latexrelease redefinitions from ltcmd.dtx . . . . .	75	
2022-02-28 ltvers.dtx v1.1i		
\new@moduledate: Detect a missing \IncludeInRelease{0000/00/00}. . . . .	42	
2022-03-10 ltbibl.dtx v1.1t		
\cite: Ensure that an empty argument generates a warning (gh/790) . . . . .	783	
2022-03-10 ltfilehook.dtx v1.0m		
\@curr@file@reqd: Add \set@curr@file@nosearch for graphicx . . . . .	861	

# Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	
\!	b398, b400, Z509
\!	<u>I201</u>
\"	s226, s378, s419, s457, s468, s579, s611, s638, s646, s652, s656, s662, s666, s672, s678, s685, s686, s692, s696, s746, s792, s1244, s1262, s1268, s1272, s1278, s1282, s1288, s1294, s1301, s1302, s1308, s1312, s1314, s1421, v416, E171, E198, Z510
\#	a65, a78, b6, b14, b498, f696, i341, v403, Z493
\\$	a77, b4, b13, f695, s307, s444, s451, s565, s804, s811, E631, E638, Z494
\\$	<u>1011</u>
\%	a78, a108, a110, a130, b14, b496, f696, s491, s493, v405, E640, E760, H135, T1451, T1452, Z495
\&	a77, b5, b13, b497, f695, T376, Z496
\'	b518, s227, s379, s421, s455, s465, s581, s591, s597, s599, s602, s604, s612, s618, s624, s626, s629, s631, s639, s643, s650, s654, s659, s664, s667, s669, s676, s681, s682, s689, s694, s697, s747, s794, s813, s815, s816, s817, s820, s822, s823, s824, s826, s827, s1238, s1259, s1266, s1270, s1275, s1280, s1283, s1285, s1292, s1297, s1298, s1305, s1310, s1313, s1321, s1322, s1369, s1370, s1375, s1376, s1387, s1388, s1393, s1394, s1422, s1423, s1437, s1438, s1439, s1440, s1453, s1454, v415, A775, B253, E161, H576, K290, K311, L72, X584, Z511, I241
\(	<u>I332</u>
\)	<u>I258</u>
\)	b518, I333
\)	<u>I258</u>
\*	v408, T1132, T1263, T1377, T1453
\*	<u>I238</u>
\+	L72
\+	<u>1009</u>
\,	b399, b401, B506, H576, I7, I8, I40, I154, I156, I159, I184, I207, I208, I224
\,	<u>1050</u>
\-	b367, f24, p416, p478, s416, s417, s574, s788, s789, v410,
\-	H576, K289, K310, L72, Z210, Z250
\-	<u>105</u> , <u>328</u> , <u>1039</u> , <u>1045</u> , <u>1051</u> , <u>f730</u>
\.	b398, b400, r45, r114, r172, s228, s380, s452, s453, s474, s587, s588, s614, s615, s641, s748, s818, s825, s1243, s1325, s1326, s1335, s1336, s1345, s1346, s1363, s1424, s1425, s1461, s1462, v409, E173, E199
\.	<u>1038</u>
\.default	<u>435</u>
.code	<u>U4</u>
.if	<u>U4</u>
.store	<u>U4</u>
.usage	<u>U4</u>
\/	a100, f25, i199, v357, v411, T375
\/	<u>1005</u>
\:	b399, b401, f690, f691, <u>I201</u> , I211, I212, I213, I229, <u>I239</u> , I239
\;	b399, b401, B500, I185
\;	<u>640</u> , <u>I201</u>
\<	s575, s739, v406, H576, L71, L109
\=	s229, s381, s473, s749, s1241, s1315, s1316, s1331, s1332, s1354, s1355, s1356, s1381, s1382, s1407, s1408, s1441, s1442, s1443, s1444, s1459, s1460, s1467, s1468, A775, E179, K290, K311, L71
\>	s572, s740, v407, H576, L71, I213, I228, I229, I239
\?	b398, b400, Z511
@@ commands:	
\@_cmd_type_cases:Nnnn	<u>1057</u>
\@_if_file_hook:wTF	<u>1059</u>
\g @_input_file_seq	<u>1055</u>
\@_set_curr_file:nNN	<u>V387</u>
\@_set_curr_file_assign:nnmNN	<u>V387</u>
\@@textvisiblespace\_meta\_{hook}	<u>207</u>
\@@next\textvisiblespace\_meta\_{hook}	<u>207</u>
\@@par	<u>290</u>
\@TeXversion	<u>1, 6</u>
@botlist commands:	
\@botlist:	X1029, X1123, X1282
@botnum commands:	
\@botnum:	X1003
\@car	<u>80</u>
\@cdr	<u>80</u>

**File Key:** a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

**File Key:** a=ltdirname.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltppageno.dtx, G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

s1296, s1304, s1309, v414, A775,  
 E175, H576, K290, K311, L72, Z514  
 \l . . . . . s561, t169, t180, B573, B574, Z515  
 \~ . . . . . a78, b10, b14, f696, l20,  
 p422, s239, s287, s384, s467, s559,  
 s645, s657, s661, s671, s687, s691,  
 s752, s1240, s1257, s1261, s1273,  
 s1277, s1287, s1303, s1307, s1351,  
 s1352, s1353, s1405, s1406, E187,  
 E207, H507, H522, H537, H580, Z502

**A**

\A . . . . . Z238, Z517, Z538  
 \a . . . . . 1008, 1009,  
 1021, 1028, s223, L1, Z229, Z518, Z529  
 \AA . . . . . 1033, 1034, b410, s240, s428, s526  
 \aa . . . . . 1033, 1034, b410, s245, s422, s536  
 \abovedisplayshortskip . . . . . b385, I501  
 \abovedisplayskip . . . . . 1019,  
 b384, I494, I496, I498, I499, I500, I501  
 \accent . . . . . 447,  
 1036, 1049, s73, s393, s423, s479, s763  
 \ActivateGenericHook . . . . .  
 . . . . . 182, 197, h1401, h1407  
 \active . . . . . a67, a122,  
 a333, b10, b11, b424, b425, b433,  
 b434, b446, b447, b451, H406,  
 H407, H506, H512, H521, H527,  
 H536, H578, T1133, T1134, T1135,  
 T1216, T1219, T1222, T1264,  
 T1265, T1266, T1348, T1351,  
 T1354, T1378, T1379, T1380,  
 T1438, T1441, T1444, X584, I241, I256  
 \acute . . . . . B516  
 add commands:  
   add\_to\_callback . . . . . d711  
 \add\_to\_callback . . . . . 47  
 \addcontentsline . . . . .  
   . . . . . 619, 1018, O70, O80, O159, P16  
 \AddEverypageHook . . . . . 878  
 \Adding . . . . . 994  
 \addpenalty . . . . . 322, p270, J124,  
 J170, J175, O50, X338, X1153, X1319  
 \AddThispageHook . . . . . 878  
 \addtocontents . . . . .  
   . . . . . 619, 1018, O164, O171, O177, O180  
 \addtocounter . . . . . 397  
 \addtocounter . . . . . 1006, t6, t18  
 \AddToHook . . . . . 181, 183, 185, 189, 193,  
 196–198, 200, 208, 209, 243, 255,  
 259, 851, 876–878, 1053, h1410,  
 h1540, x152, H38, H39, H303,  
 H304, H305, H306, R72, T1038,

T1039, T1040, V558, V560, V570,  
 V571, V572, V573, W188, W462, W485  
 \AddToHookNext . . . . . 184,  
 185, 189, 193, 239, 869, 874, 877,  
 878, 1053, h1412, h1538, V559, W486  
 \addtolength . . . . . 405  
 \addtolength . . . . . u16, I496, I498  
 \addtoversion . . . . . y20, y139  
 \addvspace . . . . . 1005, p233, H336,  
 J124, J171, J172, J176, J224, O50  
 \adjdemerits . . . . . b332  
 \AE . . . . . s241, s398,  
 s527, s768, s1127, s1437, s1441, Z568  
 \ae . . . . . s246, s401,  
 s537, s772, s1133, s1439, s1443, Z568  
 \afterassignment . . . . . 91, b468, b471,  
 f256, f262, f305, s212, s220, v328, I186  
 \AfterEndEnvironment . . . . . 200, H299  
 \aftergoup . . . . . 491  
 \aftergroup . . . . .  
   1014, 1015, v91, v342, x202, x268,  
 z114, z121, z129, D64, H510, H525,  
 H540, K148, X604, X605, X662, X663  
 \AfterLastShipout . . . . . V570  
 \afterpreamble . . . . . 334  
 \aleph . . . . . B308  
 \allocationnumber . . . . . 302, b37, b57,  
 b69, b71, b143, b144, b145, b195,  
 b196, b237, b238, b239, b252, b253,  
 b254, b271, b277, b283, b284, b297,  
 b298, b299, d52, d53, d54, d91,  
 d213, n39, L4, L9, W34, Z44, Z45, Z46  
 \allowbreak b475, f772, f773, f792, f794, I40  
 \Alph . . . . . 397  
 \Alph . . . . . 875, t140  
 \alph . . . . . 397  
 \alph . . . . . t139  
 \alpha . . . . . B268  
 \amalg . . . . . B379  
 \AmSfont . . . . . 444  
 \and . . . . . 743  
 \and . . . . . O14, O27  
 \angle . . . . . B337  
 \approx . . . . . B422  
 \arabic . . . . . 397  
 \arabic . . . . . 397, 875, t79, t87, t136, N33, W348  
 \arccos . . . . . I13  
 \arcsin . . . . . I10  
 \arctan . . . . . I16  
 \arg . . . . . I26  
 \ArgumentSpecification . . . . .  
   . . . . . 166, g2060, g2067, g2083, g2090  
 array (env.) . . . . . L168  
 \array . . . . . L168

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\arraycolsep ..... L258, L338, I360, I361, I506, I507  
 \arrayrulewidth ..... L324, L338, L346, L347, L359, L363, L366, L376, L378  
 \arraystretch ..... L186, L187, L342  
 \Arrowvert ..... B569  
 \arrowvert ..... B567  
 \asciispace ..... H466, H468, H471, H474, H475, H494  
 \ast ..... B232, B395  
 \asymp ..... B449  
 \AtBeginDocument ..... 183, 187, 196, 198, 201, 453, 785, 1030, r125, r180, w737, R54, T1029  
 \atbegindocumenthook ..... 334  
 \AtBeginDvi ..... 874, 877, 885, 886, 893, 1019, 1048, W401, W437, X86  
 \AtBeginEnvironment ..... 200, H299  
 \AtBeginShipout ..... 877, 1059, W442, W485  
 \AtBeginShipoutAddToBox ..... 876, W490  
 \AtBeginShipoutAddToBoxForeground ..... 876, W490  
 \AtBeginShipoutBox ..... 876, W483  
 \AtBeginShipoutDiscard ..... 877, W489  
 \AtBeginShipoutFirst ..... 877, W445, W487  
 \AtBeginShipoutInit ..... 877, W484  
 \AtBeginShipoutNext ..... 877, 1059, W443, W485  
 \AtBeginShipoutOriginalShipout ..... 877, W498  
 \AtBeginShipoutUpperLeft ..... 876, W490  
 \AtBeginShipoutUpperLeftForeground ..... 876, W490  
 \AtEndAfterFileList ..... V572  
 \AtEndDocument ..... 198, 201, 202, 1037, H69, T1029, V576  
 \AtEndDvi ..... 874, 877, 1059, W452, W457  
 \AtEndEnvironment ..... 200, H299  
 \AtEndOfClass ..... 850, T1029, I431  
 \AtEndOfPackage ..... 850, T541, T560, T658, T669, T1029, U49  
 \AtEndPreamble ..... 201  
 \AtNextShipout ..... 877  
 \atopwithdelims ..... I57, I58, I59  
 \attribute ..... 44, d79  
 \attributedef ..... d79, d223  
 \attributezero ..... d223  
 \AtVeryEndDocument ..... V571  
 \AtVeryVeryEnd ..... V573  
 \author ..... 743  
 \author ..... O8, O24, O32  
 B  
 \b ..... 1021, s233, s389, s475, s759, s1248  
 \backslash ..... B251, B590  
 \bar ..... B520  
 \baselineskip ..... 993, 1008, 1030, b403, b466, b502, x186, x187, x188, x190, x191, B510, K299, K318, L198, M136, M314, M373, W220, W271, X242, X273, X622, X637, X681, X696, I158, I159, I178, I184, I188  
 \baselinestretch ..... 992, v319, x120, x121, x166, x167, x184, x245  
 \baselinestretch ..... 1012  
 \batchmode ..... 348, e89, e95, e105, r602, r633, r634, y106, Z588, Z609  
 \BeforeBeginEnvironment ..... 200, H299  
 \BeforeClearDocument ..... V574  
 \begin ..... 90, 116, 140, 181, 199–201, 224, 332, 623, 996, 1055, g1213, g1319, h1429, i65, l246, l248, r340, x7, B4, B102, C4, H167, H168, M461, O14, O17, T679, U191, W383, Y3, Z475, Z479, I436, I448  
 \begingroup ..... 332, 358, 623, 992, 993, 996, 1005, 1030, 1033, 1034  
 \belowdisplayshortskip ..... b387, I500  
 \belowdisplayskip ..... b386, I499  
 \beta ..... B269  
 \bezier ..... 707  
 \bezier ..... M682, M683, M805, M806, M821, M823  
 \bf ..... 1000  
 \bfdefault ..... 203, 447, 516, 517, 519, 520, 522, 524, 527, 1049, 1050, 1058, A15, A261, A267, A268, A269, A270, A310, A311, A312, A313, A322, A358, A382, A401, A442, A476, B92, B104, B106, B114  
 \bfseries ..... 203, 516, 520, 531, A13, A14, A253, A254, A302, A307, A308, A349, A352, A353, A378, A380, A381, A474, A475, D19, G13, N36, N38, R40, W384  
 bfseries ..... A420  
 bfseries/defaults ..... A420  
 \bgroup ..... 993, 1005, 1007, 1030, b417  
 \bibcite ..... 1025, 1026, R7, R9, R10  
 \bibdata ..... R45, R49  
 \bibitem ..... R3  
 \bibliography ..... 782  
 \bibliography ..... R47  
 \bibliographystyle ..... 782  
 \bibliographystyle ..... R52  
 \bibstyle ..... R45, R57  
 \Big ..... B623, B626, B635, B637, I44, I45, I46  
 \big ..... B624, B636, I41

**File Key:** a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\bigbreak ..... b482, f774, f795  
 \bigcap ..... B345  
 \bigcirc ..... B392  
 \bigcup ..... B346  
 \Bigg ..... B630, B639, I50, I51, I52  
 \bigg ..... B628, B638, I47, I48, I49  
 \Biggl ..... I50  
 \biggl ..... I47  
 \Biggm ..... I51  
 \biggm ..... I48  
 \Biggr ..... I52  
 \biggr ..... I49  
 \Bigl ..... I44  
 \bigl ..... I41  
 \Bigm ..... I45  
 \bigm ..... I42  
 \bigodot ..... B353  
 \bigoplus ..... B352  
 \bigotimes ..... B351  
 \Bigr ..... I46  
 \bigr ..... I43  
 \bigskip ..... b487, p400  
 \bigskipamount ..... b486, p402, p403, P391  
 \bigsqcup ..... B356  
 \bigtriangledown ..... B361, B362  
 \bigtriangleup ..... B360, B363  
 \biguplus ..... B344  
 \bigvee ..... B342  
 \bigwedge ..... B343  
 \binoppenalty ..... b323  
 \bmod ..... I35  
 \boldmath ..... q14, A613  
 bool commands:  
   \bool\_gset\_false:N ..... h15, W15, W81, W90  
   \bool\_gset\_true:N ..... h10, W10, W121, W342  
   \bool\_if:NTF ..... 178,  
     g96, g107, g119, g123, g132, g134,  
     g144, g153, g154, g157, g162, g164,  
     g250, g394, g403, g405, g460, g474,  
     g487, g537, g570, g614, g624, g657,  
     g665, g670, g671, g679, g701, g714,  
     g782, g828, g829, g883, g896, g913,  
     g933, g1713, g1947, g2233, g2244,  
     h21, h1004, h1013, i159, i163, i175,  
     i297, U66, U158, W21, W88, W363  
   \bool\_lazy\_and:nnTF ..... g653,  
     h132, h1006, h1229, W66, W112, W365  
   \bool\_lazy\_and\_p:nn ..... h1232  
   \bool\_lazy\_any:nTF ..... g414  
   \bool\_lazy\_or:nnTF ..... g69, h541  
 \bool\_new:N ..... g16, g18, g20,  
     g31, g32, g34, g36, g39, g42, g43,  
     g44, h6, h24, U23, W6, W186, W201  
 \bool\_set\_false:N ..... g53, g79,  
     g188, g231, g381, g382, g383, g384,  
     g385, g406, g468, g505, g519, g541,  
     g542, g543, g558, g620, g663, g674,  
     g675, g691, g692, g694, g698, g705,  
     g833, g834, g1603, g2033, h996, U48  
 \bool\_set\_true:N .....  
     g58, g189, g220, g380,  
     g404, g480, g493, g494, g666, g667,  
     g721, g722, g728, g729, g735, g736,  
     g850, g868, g1608, g2039, h993, U46  
 \bool\_while\_do:nn ..... h800  
 \c\_false\_bool 178, 262, g1948, g2205,  
     g2503, g2608, i140, i150, i288, U139  
 \c\_true\_bool ..... 178, 262,  
     g1949, g2207, g2504, g2606, i145, U137  
 \BooleanFalse ..... g1596, g1941, g2503  
 \BooleanTrue ..... g1595, g1940, g2503  
 \bordermatrix ..... I172  
 \bot ..... B321  
 \botfigrule ..... X742, X2359  
 \botmark ..... S50, X647, X706  
 \bottomfraction ..... P275, X2328  
 \bowtie ..... B482  
 \Box ..... A726  
 \box ..... 295, 651  
 box commands:  
   \box\_dp:N ..... W192  
   \box\_gclear:N ..... n47  
   \box\_gset\_to\_last:N ..... n14, n75  
   \box\_ht:N ..... 881, W191, W300  
   \box\_if\_empty:NTF ..... W78, W97  
   \box\_if\_horizontal:NTF ..... W232, W284  
   \box\_if\_vertical:NTF ..... W206, W257  
   \box\_move\_up:nn ..... W246, W300  
   \box\_new:N n42, W23, W25, W185, W202  
   \box\_set\_dp:Nn ..... W219, W228,  
     W245, W270, W281, W299, W330  
   \box\_set\_eq:NN ..... W148, W177, W182  
   \box\_set\_eq\_drop:NN ..... W93  
   \box\_set\_ht:Nn ..... W218, W227,  
     W244, W269, W280, W298, W329  
   \box\_set\_wd:Nn .....  
     W217, W243, W268, W297  
   \box\_use:N ..... W125,  
     W223, W248, W276, W301, W331  
   \box\_use\_drop:N ..... n44  
   \box\_wd:N ..... W193, W294, W302  
 \boxmaxdepth ..... b378,  
     M475, M503, M533, M611, M628,  
     X486, X506, X546, X715, X724, X764

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltphphen.dtx, Z=ltfinal.dtx

\brace ..... I59  
 \braceleft .. B553, B557, B558, B560, B562  
 \bracerel .. B555, B559, B561  
 \bracerd .. B554, B559, B561  
 \braceru .. B556, B558, B562  
 \bracevert .. B608  
 \brack .. I58  
 \break .. 1026, b475, b480, f775, f796, p115  
 \breve .. B521  
 \brokenpenalty .. b328, v666  
 \buildrel .. B469, I149  
 \bullet .. B381

**C**

\c .. 1029, 1035, 1039, 1042,  
 s234, s335, s337, s339, s341, s343,  
 s345, s347, s349, s351, s372, s374,  
 s392, s459, s478, s606, s608, s633,  
 s635, s648, s674, s701, s704, s705,  
 s706, s707, s708, s709, s710, s711,  
 s712, s762, s1250, s1264, s1290,  
 s1347, s1348, s1367, s1368, s1371,  
 s1372, s1377, s1378, s1389, s1390,  
 s1397, s1398, s1401, s1402, E167, E196  
 \cal .. A776  
 call commands:  
     call\_callback .. d692  
 \call\_callback .. 48  
 callback commands:  
     callback\_descriptions .. d822  
 callback.register .. d576  
 \callback\_descriptions .. 48  
 \cap .. B372  
 \capitalacute ..  
     s839, E160, E161, E193, E690, E943  
 \capitalbreve ..  
     s846, E162, E163, E194, E691, E950  
 \capitalcaron ..  
     s845, E164, E165, E195, E692, E949  
 \capitalcedilla ..  
     s832, E166, E167, E196, E693, E940  
 \capitalcircumflex ..  
     s840, E168, E169, E197, E694, E944  
 \capitaldieresis ..  
     s842, E170, E171, E198, E695, E946  
 \capitaldotaccent ..  
     s848, E172, E173, E199, E696, E952  
 \capitalgrave ..  
     s838, E174, E175, E200, E697, E942  
 \capitalhungarumlaut ..  
     s843, E176, E177, E201, E698, E947  
 \capitalmacron ..  
     s847, E178, E179, E202, E699, E951

\capitalnewtie .. 582, s852,  
 E190, E191, E203, E700, E1017, E1018  
 \capitalogonek ..  
     s835, E180, E181, E204, E701, E941  
 \capitalring ..  
     s844, E182, E183, E205, E702, E948  
 \capitaltie .. s850,  
 E184, E185, E206, E703, E1013, E1014  
 \capitaltilde ..  
     s841, E186, E187, E207, E704, E945  
 \caption .. P4  
 \cases .. I153, I154, I164, I166  
 \catcode .. 619  
 \catcodetable .. 44, d89, d109  
 \catcodetable@atletter .. 45  
 \catcodetable@initex .. 45  
 \catcodetable@latex .. 45  
 \catcodetable@string .. 45  
 \catencoding .. 996  
 \cdot .. B394  
 \cdotp .. B502, B508  
 \cdots .. B508  
 center (env.) .. H351  
 \center .. H351  
 \centering .. H351,  
 H356, H357, H375, H377, H392, H394  
 \centerline .. K489  
 \changes .. 75, 1036, A667  
 \char .. 1029, f737,  
     f752, s391, s394, s430, s433, s444,  
     s451, s477, s481, s486, s489, s491,  
     s493, s733, s761, s764, s797, s804,  
     s811, s834, s837, s866, s896, s1006,  
     s1041, s1165, s1167, s1169, s1216,  
     A626, A633, E631, E638, E640,  
     E760, H466, H581, M232, M282,  
     M296, M304, M307, M448, M563,  
     M568, M576, M580, M616, M617,  
     M619, M632, M633, M636, M663, I238

char commands:

- \char\_generate:nn .. e142, g1425, g1721
- \char\_set\_catcode\_active:N .. g1972
- \char\_set\_catcode\_active:n .. g2096
- \char\_set\_catcode\_escape:N .. i338
- \char\_set\_catcode\_group\_begin:N .. i339
- \char\_set\_catcode\_group\_end:N .. i340
- \char\_set\_catcode\_other:N .. g1710
- \char\_set\_catcode\_other:n .. g1714
- \char\_set\_catcode\_parameter:N .. i341
- \char\_set\_catcode\_parameter:n .. g1715
- \char\_set\_lccode:nn .. g1977, g2106
- \char\_value\_catcode:n .. i292

\chardef .. 1002, 1009, 1044,  
 a67, a73, a74, b10, b16, b17, b18,

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

b19, b20, b58, b64, b66, b73, b79,  
 b82, b84, b94, b96, b97, b98, b99,  
 b108, b114, b115, b128, b130, b194,  
 b253, b257, b259, b283, b298, b496,  
 b497, b498, d22, d26, d38, d47,  
 d48, d89, d157, d224, j2, r52, r121,  
 s18, v14, L4, L9, T1139, T1270,  
 T1387, Z28, Z30, Z34, Z53, Z157,  
 Z158, Z159, Z160, Z161, Z162, Z163  
 $\backslash$ charsubdef ..... Z339  
 $\backslash$ charzero ..... d224  
 $\backslash$ check ..... B522  
 $\backslash$ CheckCommand ..... 1013, f187  
 $\backslash$ CheckEncodingSubset . 576, E16, E61,  
 E109, E110, E111, E156, E158,  
 E352, E621, E832, E882, E938,  
 E939, E1007, E1124, E1127, E1141  
 $\backslash$ chi ..... B288  
 $\backslash$ choose ..... I57  
 $\backslash$ circ ..... B391  
 $\backslash$ circle ... 996, M450, M605, M807, M824  
 $\backslash$ citation ..... R11, R39, R67, R84  
 $\backslash$ cite ..... 782  
 $\backslash$ cite ..... 783, R12  
 $\backslash$ clap ..... 1050, K493  
 class/.../after ..... 850  
 class/.../before ..... 850  
 class/after ..... 850  
 class/before ..... 850  
 $\backslash$ ClassError ..... l84  
 $\backslash$ ClassInfo ..... l84  
 $\backslash$ ClassNote ..... 1057, l136  
 $\backslash$ ClassNoteNoLine ..... l136  
 $\backslash$ ClassWarning ..... l84  
 $\backslash$ ClassWarningNoLine ..... l84  
 $\backslash$ cleaders ..... b516, B548, B551  
 $\backslash$ cleardoublepage ..... X138  
 $\backslash$ ClearHookNext ..... 185, h1414  
 $\backslash$ ClearHookRule ..... 189, h1451, h1554  
 $\backslash$ clearpage ..... 202,  
 340, 850, 851, 891, r296, r323, r327,  
 r351, r369, H17, H72, H165, X125,  
 X138, X143, X200, X407, X410,  
 X414, X455, X461, X2205, X2222  
 $\backslash$ cline ..... L367  
 clist commands:  
 $\backslash$ clist\_clear:N ..... U30  
 $\backslash$ clist\_gclear:N ..... h799  
 $\backslash$ clist\_gput\_left:Nn ..... h745  
 $\backslash$ clist\_gput\_right:Nn ..... h747  
 $\backslash$ clist\_if\_empty:NTF ..... h1023  
 $\backslash$ clist\_map\_inline:Nn ..... U81, U93, U146  
 $\backslash$ clist\_map\_inline:nn ..... h572, h580, U177  
 $\backslash$ clist\_new:N ..... h86, U22  
 $\backslash$ clist\_put\_right:Nn .....  
 .... U21, U77, U85, U86, U97, U108  
 $\backslash$ clist\_remove\_all:Nn ..... U98  
 $\backslash$ clist\_use:Nn ..... h1025  
 $\backslash$ clubpenalty ..... 301, 332, 1032, 1042,  
 b325, r8, r25, r94, r151, v664, J128,  
 J194, J196, O100, O106, O130, O135  
 $\backslash$ clubsuit ..... B331  
 cmd internal commands:  
 $\backslash$ \_\_cmd\_add\_arg:n ..... 146, g1361,  
 g1372, g1377, g1378, g1412, g1500,  
 g1507, g1582, g1595, g1596, g1669,  
 g1727, g1734, g1747, g1747, g1752  
 $\backslash$ \_\_cmd\_add\_arg\_spec:n .....  
 .... g503, g517, g577, g651, g651, g686  
 $\backslash$ \_\_cmd\_add\_arg\_spec\_mandatory:n .  
 .... g549, g559, g565, g651, g677  
 $\backslash$ \_\_cmd\_add\_default: .....  
 .... 129, g743, g781, g798,  
 g844, g847, g854, g864, g931, g940  
 $\backslash$ \_\_cmd\_add\_default:n .. 129, g750,  
 g790, g806, g844, g844, g861, g875  
 $\backslash$ \_\_cmd\_add\_default\_E:nn .....  
 .... g758, g844, g859, g909  
 $\backslash$ \_\_cmd\_add\_expandable\_grabber:nn  
 .... g888,  
 g901, g912, g932, g942, g949, g949  
 $\backslash$ \_\_cmd\_add\_expandable\_type\_+:w ..... g866  
 $\backslash$ \_\_cmd\_add\_expandable\_type\_D:w ..  
 .... g871, g871  
 $\backslash$ \_\_cmd\_add\_expandable\_type\_D\_- aux:NN ..... g871, g877, g894  
 $\backslash$ \_\_cmd\_add\_expandable\_type\_D\_- aux:NNNN ..... g871, g878, g881  
 $\backslash$ \_\_cmd\_add\_expandable\_type\_D\_- aux:NNNn .... g871, g872, g873, g937  
 $\backslash$ \_\_cmd\_add\_expandable\_type\_E:w ..  
 .... g907, g907  
 $\backslash$ \_\_cmd\_add\_expandable\_type\_E\_- aux:n ..... g907, g911, g923  
 $\backslash$ \_\_cmd\_add\_expandable\_type\_m:w ..  
 .... g929, g929  
 $\backslash$ \_\_cmd\_add\_expandable\_type\_R:w ..  
 .... g936, g936  
 $\backslash$ \_\_cmd\_add\_expandable\_type\_t:w ..  
 .... g938, g938  
 $\backslash$ \_\_cmd\_add\_grabber:N .....  
 .... 131, g744, g751, g764,  
 g783, g791, g799, g807, g821, g821  
 $\backslash$ \_\_cmd\_add\_type\_!:\_w ..... g725  
 $\backslash$ \_\_cmd\_add\_type\_+:\_w ..... g718  
 $\backslash$ \_\_cmd\_add\_type\_>:\_w ..... g732  
 $\backslash$ \_\_cmd\_add\_type\_b:\_w ..... g740, g740

**File Key:** a=ltdirname.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

```

__cmd_add_type_D:w g747, g747
__cmd_add_type_E:w g755, g755
__cmd_add_type_m:w g779, g779
__cmd_add_type_R:w g787, g787
__cmd_add_type_t:w g795, g795
__cmd_add_type_v:w g803, g803
__cmd_allowed_token_check:N ...
 .. g501, g513, g534, g555, g593, g593
__cmd_arg_spec_tl 108, 122-
 124, g11, g88, g379, g466, g535, g668
__cmd_args_i_tl
 108, g13, g259, g265, g270, g271, g273
__cmd_args_ii_tl 108,
 g14, g269, g271, g273, g311, g316, g323
__cmd_args_process:
 129, g249, g309, g309
__cmd_args_process_aux:n
 g309, g322, g326
__cmd_args_process_loop:nn
 g309, g315, g318
__cmd_args_tl
 108, 111, 118, 146, 151, g12, g212,
 g237, g254, g259, g265, g284, g313,
 g316, g329, g330, g1321, g1322,
 g1327, g1330, g1478, g1512, g1749
__cmd_bad_arg_spec:wn
 g436, g441, g450,
 g459, g473, g486, g500, g510, g511,
 g516, g527, g533, g554, g645, g645
__cmd_bad_def:wn
 137, g392, g400, g429, g464,
 g478, g491, g574, g582, g590, g609,
 g618, g630, g645, g650, g661, g683
__cmd_bool_reverse:N
 g1945, g1945, g2639
__cmd_break_point:n g91, g93,
 g93, g645, g650, g998, g1005, g1176
__cmd_cant_copy:nwn g995,
 g1005, g1005, g1082, g1145, g1174
__cmd_check_definable:nTF
 g2093, g2093,
 g2507, g2520, g2533, g2538, g2563,
 g2576, g2589, g2597, g2646, g2652
__cmd_check_definable_aux:nN
 168, g2093, g2094, g2097
__cmd_check_end:n g1141, g1143, g1149
__cmd_check_end:Nn
 .. g1128, g1129, g1141, g1141, g1220
__cmd_check_end:w g1141, g1151, g1154
__cmd_chk_if_free_cs:N
 g2065, g2656, g2657
__cmd_cmd_if_xparse:NTF 1056
__cmd_cmd_if_xparse_aux:N ... g2160
__cmd_cmd_type_cases:Nnnnn .. g2160
__cmd_cmd_type_cases:NnnnnTF ...
 . 137, 142, g990, g1169, g2160, g2190
__cmd_copy:NN
 . 138, g983, g985, g985, g1019, g1159
__cmd_copy_command:nNNN
 138, g991, g1020, g1020
__cmd_copy_command:NnNNnnnn
 138, g1020, g1025, g1027
__cmd_copy_environment:nnNN
 g993, g1115, g1115
__cmd_copy_environment:Nnnnnnn ...
 g1115, g1121, g1124
__cmd_copy_environment_end:nnNN ...
 g994, g1126, g1126
__cmd_copy_environment_end_-
 aux:nnNN g1126, g1130, g1133
__cmd_copy_expandable:nnN
 g1060, g1065,
 g1068, g1095, g1105, g1111, g1113
__cmd_copy_expandable:nnNN
 g992, g1033, g1033
__cmd_copy_expandable:NnNNNNnnn ...
 138, g1033, g1041, g1043
__cmd_copy_expandable_signature:NnNNNNnnn ...
 138, g1039, g1060, g1060
__cmd_copy_grabber_{type}:w ... 139
__cmd_copy_grabber_D:w
 g1085, g1085, g1098, g1099
__cmd_copy_grabber_D_alt:w
 g1085, g1097, g1100
__cmd_copy_grabber_E:w
 g1085, g1101, g1107
__cmd_copy_grabber_E_long:w
 g1085, g1107
__cmd_copy_grabber_m:w
 g1085, g1113, g1114
__cmd_copy_grabber_m_long:w
 g1085, g1114
__cmd_copy_grabber_R:w g1085, g1099
__cmd_copy_grabber_R_alt:w
 g1085, g1100
__cmd_copy_grabber_t:w g1085, g1108
__cmd_copy_parse_grabber:w
 139, g1060, g1072, g1076
__cmd_current_arg_int
 108, 123, 129, 140, g15,
 g25, g104, g133, g140, g202, g283,
 g287, g292, g293, g377, g388, g411,
 g467, g481, g495, g520, g690, g849,
 g856, g1063, g1071, g1089, g1093,
 g1094, g1104, g1227, g1293, g1302
__cmd_declare_cmd:Nnn
 . g51, g51, g2515, g2523, g2534, g2539

```

**File Key:** a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

```

__cmd_declare_cmd_aux:Nnn
..... g51, g54, g59, g61
__cmd_declare_cmd_code:Nnn
..... g89, g94, g94
__cmd_declare_cmd_code_aux:Nnn ..
..... g94, g98, g100
__cmd_declare_cmd_code_expandable:Nnn ..
..... g94, g97, g128
__cmd_declare_cmd_internal:Nnnn
..... 115, g51, g80, g82, g195
__cmd_declare_env:nnnn
..... g174, g174, g2548, g2554, g2558, g2560
__cmd_declare_env_internal:nnnn
..... g174, g190, g193
__cmd_declare_expandable_-
cmd:Nnn
..... g51, g56, g2571, g2579, g2592, g2598
__cmd_defaults: g248, g256, g256
__cmd_defaults_aux:
..... g256, g260, g261, g262, g267
__cmd_defaults_bool
..... 109, g16, g119, g134, g157, g694, g850
__cmd_defaults_def: g256, g258, g280
__cmd_defaults_def:nn
..... g256, g285, g290
__cmd_defaults_def:nnm
..... g256, g293, g295
__cmd_defaults_error:w
..... g256, g263, g275
__cmd_defaults_t1
..... 109, g17, g120, g141,
g240, g248, g284, g695, g851, g857
__cmd_delimiter_check:nnm
..... g548, g557, g634, g634
__cmd_end_expandable:NNw
..... g336, g338, g338
__cmd_end_expandable_aux:nNNNN .
..... g338, g341, g342
__cmd_end_expandable_aux:w ...
..... g338, g339, g340
__cmd_end_expandable_defaults:nnnNN
..... 119, g338, g345, g354, g363, g373, g374
__cmd_end_expandable_defaults:nnw
..... g338, g362, g366
__cmd_end_expandable_defaults:nw
..... g338, g369, g370, g372
__cmd_environment_bool ... 109,
g18, g79, g107, g189, g220, g231,
g250, g405, g570, g2033, g2039, g2244
__cmd_environment_or_command: ...
..... g278, g391, g423,
g427, g477, g490, g573, g580, g589,
g604, g648, g682, g1580, g1724,
g1731, g2049, g2053, g2242, g2242
__cmd_environment_str
..... 109, g19, g110, g176, g177,
g178, g179, g182, g186, g191, g219,
g222, g223, g251, g2040, g2042, g2245
__cmd_expandable_aux_name_t1 ..
..... 109, g21, g22, g886, g890, g899, g903
__cmd_expandable_bool 109, g20,
g53, g58, g132, g144, g188, g394,
g403, g460, g614, g624, g657, g701
__cmd_expandable_grab_D:nnNNNwNN
..... g1753, g1777, g1783
__cmd_expandable_grab_D:NNNwNNn
..... g1753, g1754, g1757
__cmd_expandable_grab_D:NNNwNNnm
..... 159, g1753, g1768, g1775, g1801, g1896
__cmd_expandable_grab_D:Nw ...
..... g1753, g1779, g1782
__cmd_expandable_grab_D:w ...
..... g1753, g1753
__cmd_expandable_grab_D_-
alt:NNwn g1820, g1827, g1923
__cmd_expandable_grab_D_-
alt:NNwNNn ... g1805, g1806, g1809
__cmd_expandable_grab_D_alt:Nwn
..... g1805
__cmd_expandable_grab_D_alt:w ..
..... g1805, g1805
__cmd_expandable_grab_E:w ...
..... g1840, g1840
__cmd_expandable_grab_E_aux:w ...
..... g1840, g1841, g1843, g1844, g1872
__cmd_expandable_grab_E_end:nnw
..... g1840, g1856, g1873
__cmd_expandable_grab_E_-
find:nnw g1840, g1870, g1871
__cmd_expandable_grab_E_find:w .
..... g1840, g1861, g1869
__cmd_expandable_grab_E_long:w .
..... g1840, g1842
__cmd_expandable_grab_E_-
loop:nnnNNw
..... g1840, g1848, g1852, g1863
__cmd_expandable_grab_E_-
test:nnw g1840, g1845, g1846
__cmd_expandable_grab_m:w
..... 140, g1875, g1875
__cmd_expandable_grab_m_aux:wNn
..... g1875, g1876, g1878, g1879
__cmd_expandable_grab_m_long:w .
..... g1875, g1877
__cmd_expandable_grab_R:w
..... g1881, g1881
__cmd_expandable_grab_R_alt:w ..
..... g1908, g1908

```

**File Key:** a=ltdirname.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfnctcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=lxref.dtx, H=ltmisenc.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

```

__cmd_expandable_grab_R_alt_-
 aux:NNwNNn ... g1908, g1909, g1912
__cmd_expandable_grab_R_-
 aux:NNNwNNn ... g1881, g1882, g1885
__cmd_expandable_grab_t:w
 g1935, g1935
__cmd_expandable_grab_t_-
 aux:NNwn g1935, g1936, g1937
__cmd_flush_m_args: ...
 ... 129, 131,
 g701, g720, g727, g734, g742, g749,
 g757, g789, g797, g805, g810, g810
\l__cmd_fn_code_t1 109, g29, g239, g254
\l__cmd_fn_t1
 ... 109, 112, 146, g28, g85, g238,
 g1323, g1366, g1390, g1396, g1406,
 g1416, g1425, g1430, g1434, g1465,
 g1491, g1499, g1501, g1506, g1508,
 g1519, g1520, g1525, g1526, g1531,
 g1532, g1537, g1538, g1543, g1544,
 g1549, g1550, g1555, g1556, g1561,
 g1562, g1592, g1598, g2034, g2249
\l__cmd_function_t1 110, g24, g30,
 g84, g86, g103, g114, g115, g131,
 g139, g149, g152, g156, g158, g166,
 g172, g399, g463, g617, g629, g660
__cmd_get_arg_spec:N
 g2068, g2068, g2647
__cmd_get_arg_spec:n
 g2068, g2073, g2649
__cmd_get_arg_spec:NTF ... g2056,
 g2056, g2070, g2075, g2082, g2088
__cmd_get_arg_spec_error:N ...
 g2031, g2031, g2071, g2084
__cmd_get_arg_spec_error:n ...
 g2031, g2037, g2078, g2091
__cmd_get_arg_spec_error_aux:n .
 g2031, g2035, g2041, g2044
__cmd_get_grabber>NN
 ... 135, g925, g941, g954, g954
__cmd_get_grabber_auxi>NN
 g954, g957, g960, g968
__cmd_get_grabber_auxii>NN
 g954, g975, g978
__cmd_grab_b:w g1309, g1309
__cmd_grab_b_aux:NNw ...
 ... g1309,
 g1310, g1312, g1314, g1316, g1317
__cmd_grab_b_end:Nw
 g1309, g1320, g1325
__cmd_grab_b_long:w ... g1309, g1311
__cmd_grab_b_long_obey_spaces:w
 g1309, g1315
__cmd_grab_b_obey_spaces:w ...
 g1309, g1313
__cmd_grab_D:w g1336, g1336
__cmd_grab_D_aux:NNnN
 ... g1319, g1356, g1358, g1363, g1575
__cmd_grab_D_aux:NNnNN ...
 ... g1338, g1343, g1348, g1353, g1356, g1356
__cmd_grab_D_call:Nw ...
 ... 147, g1360, g1420, g1420, g1577
__cmd_grab_D_long:w ... g1336, g1341
__cmd_grab_D_long_obey_spaces:w
 g1336, g1351
__cmd_grab_D_nested:NNnN
 g1369, g1383, g1386
__cmd_grab_D_nested:w
 g1383, g1401, g1418
__cmd_grab_D_obey_spaces:w ...
 g1336, g1346
__cmd_grab_E:nnNN g1439,
 g1441, g1447, g1453, g1459, g1463
__cmd_grab_E:w g1439, g1439
__cmd_grab_E_finalise:
 g1439, g1472, g1488, g1495
__cmd_grab_E_long:w ... g1439, g1445
__cmd_grab_E_long_obey_spaces:w
 g1439, g1457
__cmd_grab_E_loop:NnN ...
 ... g1439, g1468, g1483, g1485, g1492
__cmd_grab_E_obey_spaces:w ...
 g1439, g1451
\l__cmd_grab_expandably_bool ...
 ... 110, 120, g31,
 g96, g380, g404, g406, g468, g505,
 g519, g541, g558, g620, g663, g714
__cmd_grab_m:w g1496, g1496
__cmd_grab_m_1:w g1510
__cmd_grab_m_2:w g1510
__cmd_grab_m_3:w g1510
__cmd_grab_m_4:w g1510
__cmd_grab_m_5:w g1510
__cmd_grab_m_6:w g1510
__cmd_grab_m_7:w g1510
__cmd_grab_m_8:w g1510
__cmd_grab_m_9:w g1510
__cmd_grab_m_aux:Nnnnnnnn ...
 ... g1510, g1510, g1519, g1525, g1531,
 g1537, g1543, g1549, g1555, g1561
__cmd_grab_m_long:w ... g1496, g1503
__cmd_grab_R:w g1569, g1569
__cmd_grab_R_aux:NNnN
 g1569, g1570, g1572, g1573
__cmd_grab_R_long:w ... g1569, g1571
__cmd_grab_t:w g1585, g1585

```

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

```

__cmd_grab_t_aux:NNw
 g1585, g1586, g1588, g1589
__cmd_grab_t_obey_spaces:w . . .
 g1585, g1587
__cmd_grab_v:w g1601, g1601
__cmd_grab_v_aux:w
 155, g1601, g1604, g1609, g1611
__cmd_grab_v_aux_abort:n . . .
 154, g1627, g1645, g1651,
 g1664, g1683, g1706, g1708, g1717
__cmd_grab_v_aux_catcodes: . . .
 153, g1642, g1674, g1708, g1708
__cmd_grab_v_aux_loop:N . . .
 g1637, g1643, g1647, g1661
__cmd_grab_v_aux_loop>NN . . .
 g1637, g1650, g1653
__cmd_grab_v_aux_loop_end: . . .
 g1637, g1658, g1666, g1697
__cmd_grab_v_aux_put:N . . .
 g1641, g1660,
 g1676, g1694, g1702, g1737, g1737
__cmd_grab_v_aux_test:N . . .
 g1626, g1637, g1637
__cmd_grab_v_bgroup: . . .
 g1622, g1672, g1672
__cmd_grab_v_bgroup_loop: . . .
 g1672, g1677, g1679, g1695, g1703
__cmd_grab_v_bgroup_loop:N . . .
 g1672, g1682, g1685
__cmd_grab_v_group_end: . . .
 154, g1601, g1631, g1668, g1719
__cmd_grab_v_long:w g1601, g1606
__cmd_grab_v_token_if_char:NTF .
 154, g1639, g1655, g1687, g1745, g1745
\g__cmd_grabber_int . . .
 109, g27, g967, g971
\c__cmd_ignore_def_tl . 171, g2240,
 g2258, g2265, g2279, g2300, g2329,
 g2336, g2343, g2351, g2359, g2368,
 g2375, g2382, g2389, g2401, g2408
\l__cmd_last_delimiters_tl . 110,
 120, 123, g33, g378, g396, g504,
 g518, g540, g576, g626, g636, g685
\l__cmd_long_bool . . .
 110, 134, g34, g382, g474,
 g480, g543, g654, g665, g670, g674,
 g691, g721, g828, g833, g868, g883,
 g896, g913, g933, g1603, g1608, g1713
\l__cmd_m_args_int 110,
 g35, g693, g784, g812, g815, g817, g819
\l__cmd_nesting_a_tl g1383
\l__cmd_nesting_b_tl g1383
__cmd_normalize_arg_spec:n . . .
 g87, g375, g375
__cmd_normalize_arg_spec_loop:n
 g375, g386, g408, g469, g482, g496,
 g506, g521, g544, g550, g560, g566
__cmd_normalize_check_gv:N . . .
 g564, g612, g612
__cmd_normalize_check_lu:N . . .
 g612, g622
__cmd_normalize_E_unique_
 check:w g498, g514, g523, g528
__cmd_normalize_type_!w g457
__cmd_normalize_type_+w g457
__cmd_normalize_type_>w g457
__cmd_normalize_type_b:w g568, g568
__cmd_normalize_type_D:w . . .
 g437, g445, g447, g498, g498
__cmd_normalize_type_d:w g432, g434
__cmd_normalize_type_E:w . . .
 g442, g498, g508
__cmd_normalize_type_e:w g432, g439
__cmd_normalize_type_m:w g546, g546
__cmd_normalize_type_0:w g432, g446
__cmd_normalize_type_o:w g432, g444
__cmd_normalize_type_R:w . . .
 g451, g546, g552
__cmd_normalize_type_r:w g432, g448
__cmd_normalize_type_s:w g432, g453
__cmd_normalize_type_t:w . . .
 g454, g498, g531
__cmd_normalize_type_v:w g546, g562
\l__cmd_obey_spaces_bool 110, 132,
 g32, g381, g487, g493, g537, g542,
 g671, g675, g692, g728, g829, g834
__cmd_peek_cs_check_equal:NNN . . .
 g2204, g2218, g2224
__cmd_peek_meaning:NTF . . .
 g2195, g2204, g2204
__cmd_peek_meaning_aux:NNTF . . .
 g2204, g2205, g2207, g2208
__cmd_peek_meaning_remove:NTF . . .
 g1349, g1354, g1455,
 g1461, g1588, g2197, g2204, g2206
__cmd_peek_nonspace:NTF g2194, g2194
__cmd_peek_nonspace_aux:nNNTF . . .
 g2194, g2195, g2197, g2198, g2201
__cmd_peek_nonspace_remove:NTF . . .
 g1339, g1344, g1443,
 g1449, g1576, g1586, g2194, g2196
__cmd_peek_true_remove:NNw g2204
__cmd_peek_true_remove:Nw . . .
 g2215, g2219, g2227, g2231
\l__cmd_prefixed_bool . . .
 110, g36, g705, g722, g729, g735, g782

```

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntr1.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

```

__cmd_prepare_signature:N
..... 129, g688, g700, g703,
g745, g753, g766, g785, g793, g801,
g808, g869, g879, g921, g934, g947
__cmd_prepare_signature:n
..... g88, g688, g688
__cmd_prepare_signature_-
bypass:N 129,
g688, g706, g708, g723, g730, g738
\l__cmd_process_all_tl
..... 110, 118, g37, g124,
g241, g249, g314, g696, g816, g835
\l__cmd_process_one_tl . 110, 131,
g38, g697, g737, g762, g771, g839, g842
\l__cmd_process_some_bool
..... 110, g39, g123, g698, g736
__cmd_put_arg_expandable:nw ...
..... g1789, g1794, g1795, g1830,
g1835, g1836, g1943, g1943, g1944
__cmd_replicate_processor:nn ...
..... g761, g768, g768
__cmd_run_code:
116, g243,
g246, g246, g1317, g1325, g1333,
g1336, g1341, g1346, g1351, g1439,
g1445, g1451, g1457, g1481, g1496,
g1503, g1514, g1516, g1522, g1528,
g1534, g1540, g1546, g1552, g1558,
g1569, g1571, g1589, g1611, g1750
\l__cmd_saved_args_tl
..... 111, g40, g1321, g1329
__cmd_set_environment_end:n ...
..... g207, g251
__cmd_set_eq_if_exist:NN
..... g985, g1002, g1004,
g1023, g1036, g1037, g1038, g1118
__cmd_show:N
143, g1162, g1164, g1164, g1224, g1306
__cmd_show_arg_spec:N
..... g2080, g2080, g2653
__cmd_show_arg_spec:n
..... g2080, g2086, g2655
__cmd_show_command:N
..... g1170, g1180, g1180
__cmd_show_command:NnNNwN
..... g1180, g1181, g1182
__cmd_show_command_aux:nNNn 142,
143, g1180, g1183, g1187, g1188, g1212
__cmd_show_delim:Nw .. g1254, g1254
__cmd_show_delims:Nw .. g1254, g1256
__cmd_show_delims_opt:Nw
..... g1254, g1258
__cmd_show_E:Nw
g1254, g1273
__cmd_show_e:Nw
g1254, g1262
__cmd_show_environment:N
..... g1172, g1180, g1198, g1221
__cmd_show_environment:Nnnw ...
..... g1200, g1208
__cmd_show_environment_end:N ...
..... g1173, g1218
__cmd_show_expandable:N
..... g1171, g1180, g1184
__cmd_show_expandable:NnNNNNnN .
..... g1180, g1185, g1186
__cmd_show_opt:Nw
g1254, g1260
__cmd_show_prefix:Nw ..
g1254, g1286
__cmd_show_processor:Nw g1254, g1288
\c__cmd_show_type_!_tl
g1248
\c__cmd_show_type_+_tl
g1248
\c__cmd_show_type_>_tl
g1248
\c__cmd_show_type_D_tl
g1248
\c__cmd_show_type_d_tl
g1248
\c__cmd_show_type_E_tl
g1248
\c__cmd_show_type_e_tl
g1248
\c__cmd_show_type_0_tl
g1248
\c__cmd_show_type_R_tl
g1248
\c__cmd_show_type_r_tl
g1248
\c__cmd_show_type_t_tl
g1248
\l__cmd_signature_tl
... 111, 146, 157, g41, g117, g160,
g699, g752, g765, g792, g800, g814,
g823, g951, g1320, g1365, g1471,
g1481, g1498, g1505, g1514, g1518,
g1524, g1530, g1536, g1542, g1548,
g1554, g1560, g1591, g1613, g1750
__cmd_single_token_check:n
..... g501, g502,
g512, g534, g555, g556, g584, g584
\l__cmd_some_long_bool
111,
120, g43, g154, g162, g384, g655, g666
\l__cmd_some_obey_spaces_bool ...
..... 111, g42, g383, g494, g679
\l__cmd_some_short_bool
111, 120, g44, g153, g164, g385, g667
__cmd_split_add_item:n
... 145, g1287, g1289, g1290, g1295
__cmd_split_argument:nnn
..... g1986, g1986, g2640
__cmd_split_argument_aux:n ...
..... g1986, g2002, g2021
__cmd_split_argument_aux:nnnn ...
..... g1986, g1989, g1993
__cmd_split_argument_aux:wn ...
..... g1986, g2022, g2023

```

**File Key:** a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\\_\_cmd\_split\_end\_item:n .....  
     . 145, g1246, g1255, g1257, g1259,  
         g1261, g1267, g1278, g1290, g1297  
 \\_\_cmd\_split\_list:nn .....  
     . . . . . g1951, g1953, g1988, g2641  
 \\_\_cmd\_split\_list\_multi:nn g1951,  
     g1958, g1961, g1963, g1970, g1983  
 \l\_\_cmd\_split\_list\_seq .....  
     . . . . . 164, g1951, g1965, g1967  
 \\_\_cmd\_split\_list\_single:Nn . . .  
     . . . . . g1951, g1959, g1973  
 \l\_\_cmd\_split\_list\_t1 .....  
     . . . . . 164, g1952, g1975, g1981, g1983  
 \\_\_cmd\_split\_signature:n .....  
     . . . . . 142, g1190, g1225, g1225  
 \\_\_cmd\_split\_signature\_loop:Nw ..  
     . . . . . g1230, g1232,  
         g1232, g1246, g1255, g1257, g1259,  
         g1261, g1269, g1282, g1287, g1289  
 \\_\_cmd\_split\_start\_item: . . . 144,  
     145, g1235, g1266, g1277, g1290, g1290  
 \\_\_cmd\_start:nNNnnn .....  
     . . . . . 113, 130, 169, g113, g215, g226, g2176  
 \\_\_cmd\_start\_aux:NNnnnn .....  
     . . . . . g221, g232, g235, g235, g245  
 \\_\_cmd\_start\_env:nnnnnn .....  
     . . . . . 169, g109, g215, g215, g2178  
 \\_\_cmd\_start\_expandable:nNNNNn ..  
     . . . . . 113, 169, g147, g333, g333, g2177  
 \\_\_cmd\_t1\_mapthread\_function:NNN  
     . . . . . g284, g312, g2134, g2134  
 \\_\_cmd\_t1\_mapthread\_function:nnN  
     . . . . . 145, g360, g1280, g2134, g2147  
 \\_\_cmd\_t1\_mapthread\_loop:w .....  
     . . . . . g2134, g2137, g2149, g2153, g2158  
 \\_\_cmd\_tmp:w .....  
     . . . . . 117, 122, 158, g45, g48, g270,  
         g286, g328, g330, g432, g456, g956,  
         g962, g980, g1147, g1157, g1248,  
         g1251, g1253, g1271, g1275, g1281,  
         g1285, g1755, g1774, g1807, g1826,  
         g1883, g1907, g1910, g1934, g2236  
 \l\_\_cmd\_tmp\_prop .....  
     . . . . . 111, g45, g1467, g1470, g1476  
 \l\_\_cmd\_tmpt1 . . . 111, 142, 143,  
     g46, g282, g287, g297, g925, g927,  
     g941, g944, g1058, g1064, g1079,  
     g1087, g1103, g1110, g1128, g1131,  
     g1194, g1220, g1221, g1228, g1299,  
     g1404, g1407, g2210, g2235, g2238  
 \l\_\_cmd\_tmptb\_t1 .....  
     . . . . . 111, 135, 143, 144, g47, g910, g915,  
         g926, g1129, g1131, g1229, g1235,  
         g1292, g1296, g1300, g1301, g1476,  
         g1477, g1479, g2211, g2222, g2228  
 \\_\_cmd\_token\_if\_cs:NTF .....  
     . . . . . 125, g1428, g2125, g2125, g2217  
 \\_\_cmd\_trim\_spaces:n .....  
     . . . . . g2029, g2029, g2642  
 \l\_\_cmd\_v\_arg\_t1 .....  
     . . . . . 153, 154, g1600, g1616,  
         g1635, g1669, g1725, g1732, g1739  
 \l\_\_cmd\_v\_nesting\_int .....  
     . . . . . 155, g1671, g1675, g1691, g1692, g1701  
 \colon ..... B503  
 \columnsep . . . r27, r96, r153, X81, X202  
 \columnseprule . . . X82, X2260, X2294  
 \columnwidth . . . r24, r27, r28, r30,  
     r93, r96, r97, r99, r150, r153, r154,  
     r157, K346, K377, K395, K412,  
     P99, P168, P470, P489, P507,  
     X80, X144, X145, X146, X201,  
     X202, X203, X204, X205, X1843,  
     X1845, X2258, X2262, X2290, X2296  
 \cong ..... B457  
 \contentsline ..... 752,  
     753, 1022, O164, O171, O177, O184  
 \coprod ..... B341  
 \copyright ..... 1017, 1021,  
     1022, 1030, 1034, s285, s316, A649  
 \cos ..... I12  
 \cosh ..... I14  
 \cot ..... I18  
 \coth ..... I19  
 \countdef ..... a69, b37,  
     b38, b39, b41, b51, b90, d75, d85,  
     d174, d190, d198, d206, d225, F3, Z61  
 \counterwithin ..... 397  
 \counterwithin ..... 1057, t75  
 \counterwithout ..... 397  
 \counterwithout ..... 1057, t75  
 \CountZero ..... d225  
 \cr ..... b413, s500, s506,  
     s516, s522, E101, E105, E930, E934,  
     L192, L203, L210, L219, L224,  
     L230, L377, M141, M143, M148,  
     M154, I175, I179, I365, I411, I510  
 \crcr ..... b503, s327, s362,  
     s363, s390, s394, s397, s476, s480,  
     s484, s486, s489, s732, s760, s764,  
     s767, s834, s837, s895, s1253, A651,  
     B337, B338, B340, B459, B462,  
     B466, B530, B531, B532, B533,  
     B534, B535, B537, B538, B539,  
     B540, B541, B543, E106, E935,  
     L171, L172, M141, I155, I157, I158,  
     I159, I175, I177, I178, I179, I197, I198

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=lt hyphen.dtx, Z=ltfinal.dtx

```

create commands:
 create_callback d667
 \create_callback 48
 \CS 80
 \cs h1421, h1422, h1424
cs commands:
 \cs:w 214, g1431, g2181, h219,
 h704, h738, h739, h791, h811, h814,
 h830, h831, h854, h855, h856, h863,
 h870, h1122, h1128, h1141, h1151,
 h1187, h1217, h1262, h1292, z187,
 z190, U119, V211, V225, V237, V238
 \cs_argument_spec:N
 263, 268, e139, g2164, i158, i287
 \cs_end: g1431, g2183,
 h219, h448, h707, h738, h739, h791,
 h811, h814, h830, h831, h845, h855,
 h856, h863, h870, h1063, h1122,
 h1127, h1128, h1141, h1147, h1160,
 h1187, h1217, h1262, h1292, z187,
 z190, U119, V211, V225, V237, V238
 \cs_generate_from_arg_count:NNnn
 g102,
 g130, g138, g200, g286, g289, g328
 \cs_generate_variant:Nn
 g245, g289, g1004,
 g1752, g1944, g1970, h35, h36, h37,
 h44, h45, h52, h53, h54, h57, h63,
 h67, h833, i19, U21, U116, V462, V470
 \cs_gset:Npn z192, V242, W171
 \cs_gset:Npx V213
 \cs_gset_eq:NN
 e136, e137, e138, e139, e140, e141,
 e142, g2066, g2656, h451, h711,
 h724, h725, h1114, i55, i384, z184,
 z198, W52, W150, W164, W165,
 W170, W179, W183, W319, W376
 \cs_gset_nopar:Npn U28
 \cs_gset_nopar:Npx h47, h49, h51
 \cs_gset_protected:Npn h1434
 \cs_gset_protected:Npx h20, W20
 \cs_if_eq:NNTF
 g962, g1487, U61, U63, U104
 \cs_if_exist:N 845
 \cs_if_exist:NTF
 865, 866, g63, g179,
 g965, g1003, g2035, g2042, g2509,
 g2522, g2534, g2543, g2546, g2553,
 g2558, g2565, g2578, g2591, h1272,
 h1280, i72, i80, U106, V93, V439, V443
 \cs_if_exist_p:N g70, g71
 \cs_if_exist_use:NTF
 e181, g412, g1081,
 h652, W311, W312, W316, W317
 \cs_if_free:NTF U73
 \cs_new:Npn
 e179, g333, g338, g340, g342, g354,
 g366, g372, g1027, g1043, g1124,
 g1149, g1154, g1418, g1753, g1757,
 g1775, g1782, g1783, g1805, g1809,
 g1827, g1840, g1842, g1844, g1846,
 g1852, g1869, g1871, g1873, g1875,
 g1877, g1879, g1881, g1885, g1908,
 g1912, g1935, g1937, g1943, g2021,
 g2023, g2134, g2147, g2153, g2242,
 g2600, g2617, g2618, g2622, g2623,
 g2624, h39, h170, h176, h189, h199,
 h200, h202, h216, h222, h559, h561,
 h563, h686, h702, h756, h757,
 h834, h835, h1039, h1125, h1145,
 h1153, h1167, h1270, h1278, h1290,
 h1440, h1441, i224, i234, i244, i246,
 i248, i259, i278, i279, i282, i409,
 n43, n46, n71, n83, U113, U115,
 V30, V38, V44, V57, V95, V100,
 V105, V112, V127, V234, V240,
 V412, V414, V416, V419, V423,
 V437, V453, V463, V579, W54,
 W59, W77, W140, W145, W161,
 W176, W181, W187, W190, W204,
 W255, W308, W321, W335, W338,
 W348, W381, W500, W501, W502
 \cs_new_eq:NN
 e159, e160, e161, e162, e178, g48,
 g93, g1099, g1100, g1107, g1114,
 g2065, g2503, g2504, g2619, g2620,
 g2621, g2627, g2628, g2629, g2639,
 g2640, g2641, g2642, g2643, g2649,
 g2655, h7, h23, h34, h64, h665,
 h671, h709, h909, h910, h911, h912,
 h913, h914, h1160, h1453, h1454,
 h1524, h1526, i12, i13, i335, n93,
 U8, V252, V253, V506, V508, V510,
 V512, V514, V516, V518, V520,
 V522, V524, W7, W151, W345,
 W347, W400, W403, W404, W499
 \cs_new_protected:Npn
 893, g51, g56, g61,
 g82, g94, g100, g128, g174, g193,
 g207, g215, g235, g246, g256, g267,
 g275, g280, g290, g295, g309, g318,
 g326, g375, g408, g434, g439, g444,
 g446, g448, g453, g457, g471, g484,
 g498, g508, g523, g531, g546, g552,
 g562, g568, g584, g593, g612, g622,
 g634, g645, g650, g651, g677, g688,
 g703, g708, g718, g725, g732, g740,
 g747, g755, g768, g779, g787, g795

```

**File Key:** a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=lt hyphen.dtx, Z=ltfinal.dtx

g803, g810, g821, g844, g854, g859,  
 g866, g871, g873, g881, g894, g907,  
 g923, g929, g936, g938, g949, g954,  
 g960, g978, g985, g1002, g1005,  
 g1020, g1033, g1060, g1068, g1076,  
 g1085, g1097, g1101, g1108, g1113,  
 g1115, g1126, g1133, g1141, g1164,  
 g1180, g1182, g1184, g1186, g1188,  
 g1198, g1208, g1218, g1225, g1232,  
 g1254, g1256, g1258, g1260, g1262,  
 g1273, g1286, g1288, g1290, g1295,  
 g1297, g1309, g1311, g1313, g1315,  
 g1317, g1325, g1336, g1341, g1346,  
 g1351, g1356, g1363, g1386, g1439,  
 g1445, g1451, g1457, g1463, g1485,  
 g1495, g1496, g1503, g1516, g1522,  
 g1528, g1534, g1540, g1546, g1552,  
 g1558, g1569, g1571, g1573, g1585,  
 g1587, g1589, g1601, g1606, g1611,  
 g1631, g1637, g1647, g1653, g1666,  
 g1679, g1685, g1708, g1717, g1737,  
 g1745, g1747, g1945, g1953, g1963,  
 g1973, g1986, g1993, g2029, g2031,  
 g2037, g2044, g2056, g2068, g2073,  
 g2080, g2086, g2093, g2097, g2125,  
 g2160, g2188, g2194, g2196, g2198,  
 g2204, g2206, g2208, g2224, g2231,  
 g2505, g2518, g2531, g2536, g2541,  
 g2551, g2557, g2559, g2561, g2574,  
 g2587, g2595, g2644, g2650, h8,  
 h13, h18, h40, h42, h46, h48, h50,  
 h55, h58, h65, h68, h70, h79, h91,  
 h100, h102, h107, h109, h123, h125,  
 h142, h147, h149, h167, h223, h232,  
 h237, h245, h269, h271, h287, h295,  
 h305, h314, h323, h325, h342, h366,  
 h373, h380, h390, h395, h400, h409,  
 h443, h445, h453, h455, h458, h460,  
 h601, h603, h636, h641, h660, h666,  
 h672, h677, h680, h683, h684, h710,  
 h727, h763, h836, h843, h852, h859,  
 h866, h873, h881, h887, h898, h915,  
 h922, h934, h939, h944, h946, h948,  
 h1041, h1056, h1061, h1070, h1086,  
 h1088, h1098, h1103, h1111, h1113,  
 h1117, h1131, h1136, h1158, h1173,  
 h1182, h1191, h1196, h1203, h1205,  
 h1442, h1443, h1444, h1445, h1455,  
 h1462, h1469, h1476, h1483, h1485,  
 h1487, h1494, h1501, h1508, h1515,  
 i17, i22, i24, i33, i35, i45, i53,  
 i62, i76, i96, i116, i129, i134, i143,  
 i148, i153, i217, i284, i319, i336,  
 i362, n51, z179, z210, U24, U26,  
 U59, U71, U91, U102, U117, U121,  
 U142, U156, V49, V62, V70, V79,  
 V208, V222, V393, V398, V421,  
 V544, W8, W13, W18, W341, W462  
\cs\_new\_protected:Npx .....  
..... g226, g1564, g1672  
\cs\_new\_protected\_nopar:Npn .....  
..... g1420, g1510  
\cs\_prefix\_spec:N ..... e138, i204  
\cs\_replacement\_spec:N .....  
..... e140, g1195, g1204  
\cs\_set:Npn ..... 146, g140,  
..... g201, g286, g328, g956, i196, i351, i368  
\cs\_set:Npx .....  
..... g169, g884, g897, V211, V225  
\cs\_set\_eq:NN ..... g203,  
..... g204, g980, g1003, g1022, g1035,  
..... g1092, g1117, g1122, g1137, g1139,  
..... g1519, g1525, g1531, g1537, g1543,  
..... g1549, g1555, g1561, g1710, g2236,  
..... h744, h745, h746, h747, h936, h941,  
..... i168, i169, i189, i190, i195, i347,  
..... i348, i364, i365, i422, n49, n50, n94,  
..... n95, n96, n98, n99, n100, n134,  
..... n135, n136, n139, z201, U49, W24,  
..... W44, W83, W95, W124, W131,  
..... W406, W408, W410, W412, W414  
\cs\_set\_nopar:Npn ..... 146  
\cs\_set\_nopar:Npx .....  
..... g145, g166, g171, g198, g209,  
..... g885, g898, g1040, g1135, h41, h43  
\cs\_set\_protected:Npn .....  
..... g104, g432, g1147, g1248, g1271,  
..... g1275, g1312, g1316, g1343, g1353,  
..... g1448, g1460, g1472, g1506, g1572,  
..... g1592, g1755, g1807, g1883, g1910,  
..... h258, h1043, h1073, i170, U5,  
..... U29, U43, U127, W46, W134, W401  
\cs\_set\_protected\_nopar:Npn .....  
..... g1310, g1314, g1338,  
..... g1348, g1442, g1454, g1499, g1570  
\cs\_set\_protected\_nopar:Npx .....  
..... g105, g145, g1024, g1120  
\cs\_to\_str:N .....  
..... 110, 214, e136, g70, g71, g84, g996,  
..... g998, g1176, g1203, g1205, g1220,  
..... g1431, g2183, i111, i132, i146, i151  
\cs\_undefine:N ..... g2633, g2634,  
..... g2635, g2657, h128, h685, i58, V227  
\cs\check@cr commands:  
\cs\_gset\_eq:NN ..... 75  
\csc ..... 121  
\csname ..... 186, 521, 623, 861, 865, 1011, 1046  
\csname\endcsname ..... 861, 863

**File Key:** a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

\cup . . . . .	B373	\cyrdelta . . . . .	s1511
\CurrentFile . . . . .	340, 818, 849, 852, 855, 857, 858, 863, 864, T848, V3, V67, V84, V166, V171, V174, V378, V382, V552, V553	\CYRDJE . . . . .	s1512
\CurrentFilePath . . . . .	849, 855, 858, 864, V3, V67, V83, V165, V379, V382	\cyrdje . . . . .	s1512
\CurrentFilePathUsed . . . . .	849, V3, V66, V81, V163, V376, V379	\CYRDZE . . . . .	s1512
\CurrentFileUsed . . . . .	818, 849, 852, 857, 864, T848, V3, V66, V82, V164, V376, V378, V551, V552	\cyrdze . . . . .	s1512
\CurrentOption . . . . .	997, 999, 1027, s1536, s1539, s1544, s1556, T14, T455, T466, T479, T480, T485, T498, T499, T501, T515, T516, T519, T533, T538, T539, T552, T557, T558, T571, T573, T579, T581, T593, T594, T595, T603, T604, T605, T898, T963, T1061, T1062, T1072	\CYRDZHE . . . . .	s1512
CurrentOption commands:		\cyrdzhe . . . . .	s1512
\CurrentOption: . . . . .	T478, T497, T514, T532, T537, T551, T556, T592, T602, T1071	\CYRE . . . . .	s1512
\CYRA . . . . .	s1508	\CYREPS . . . . .	s1513
\cyra . . . . .	s1508, s1559	\cyreps . . . . .	s1512
\CYRABHCH . . . . .	s1508	\CYREREV . . . . .	s1513
\cyrabhch . . . . .	s1508	\cyrerev . . . . .	s1513
\CYRABHCHDSC . . . . .	s1508	\CYRERY . . . . .	s1513
\cyrabhchdsc . . . . .	s1508	\cyrery . . . . .	s1513
\CYRABHDZE . . . . .	s1509	\CYRF . . . . .	s1513
\cyrabhdze . . . . .	s1508	\cyrf . . . . .	s1513
\CYRABHHA . . . . .	s1509	\CYRFITA . . . . .	s1514
\cyrabhha . . . . .	s1509	\cyrfita . . . . .	s1513
\CYRAE . . . . .	s1509	\CYRG . . . . .	s1514
\cyrae . . . . .	s1509	\cyrg . . . . .	s1514
\CYRB . . . . .	s1509	\CYRGDSC . . . . .	s1514
\cyrb . . . . .	s1509	\cyrgdsc . . . . .	s1514
\CYRBYUS . . . . .	s1510	\CYRGDSCHCRS . . . . .	s1514
\cyrbyus . . . . .	s1509	\cyrgdschcrs . . . . .	s1514
\CYRC . . . . .	s1510	\CYRGHCRS . . . . .	s1515
\cyrc . . . . .	s1510	\cyrghcrs . . . . .	s1515
\CYRCH . . . . .	s1510	\CYRGHK . . . . .	s1515
\cyrch . . . . .	s1510	\cyrghk . . . . .	s1515
\CYRCHLDSC . . . . .	s1510	\CYRGUP . . . . .	s1515
\cyrchldsc . . . . .	s1510	\cyrgup . . . . .	s1515
\CYRCHRDSCL . . . . .	s1511	\CYRH . . . . .	s1515
\cyrchrdsc . . . . .	s1510	\cyrh . . . . .	s1515
\CYRCHVCRS . . . . .	s1511	\CYRHDSCL . . . . .	s1516
\cyrchvcrs . . . . .	s1511	\cyrhdscl . . . . .	s1516
\CYRD . . . . .	s1511	\CYRHHCRS . . . . .	s1516
\cyrd . . . . .	s1511	\cyrhhcrs . . . . .	s1516
\CYRDELTA . . . . .	s1511	\CYRHKK . . . . .	s1516
		\cyrhhk . . . . .	s1516
		\CYRHRDSN . . . . .	s1517
		\cyrhrdsn . . . . .	s1516
		\CYRI . . . . .	s1517
		\cyri . . . . .	s1517
		\CYRIE . . . . .	s1517
		\cyrrie . . . . .	s1517
		\CYRII . . . . .	s1517
		\cyrii . . . . .	s1517
		\CYRISHRT . . . . .	s1517
		\cyrishrt . . . . .	s1517
		\CYRISHRTDSC . . . . .	s1518
		\cyrishrtdsc . . . . .	s1518
		\CYRIZH . . . . .	s1518
		\cyrizh . . . . .	s1518
		\CYRJE . . . . .	s1518

**File Key:** a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\cyrje .....	s1518	\cyrrhk .....	1037, s1524
\CYRK .....	s1518	\CYRRHOOK .....	1037
\cyrk .....	s1518	\cyrrhook .....	1037
\CYRKBEAK .....	s1519	\CYRRTICK .....	s1525
\cyrkbeak .....	s1519	\cyrrtick .....	s1525
\CYRKDSC .....	s1519	\CYRS .....	s1525
\cyrkdsc .....	s1519	\cyrs .....	s1525
\CYRKHCRS .....	s1519	\CYRSACRS .....	s1525
\cyrkhcrs .....	s1519	\crysacrs .....	s1525
\CYRKHK .....	s1520	\CYRSCHWA .....	s1526
\cyrkhk .....	s1519	\cryschwa .....	s1526
\CYRKVCRS .....	s1520	\CYRSDSC .....	s1526
\cyrkvcrs .....	s1520	\crysdesc .....	s1526
\CYRL .....	s1520	\CYRSEMISFTSN .....	s1526
\cyrl .....	s1520	\cyrsemisftsn .....	s1526
\CYRLDSC .....	s1520	\CYRSFTSN .....	s1527
\cyrldesc .....	s1520	\crysftsn .....	s1527
\CYRLHK .....	s1521	\CYRSH .....	s1527
\cyrlhk .....	s1520	\cyrsh .....	s1527
\CYRLJE .....	s1521	\CYRSHCH .....	s1527
\cyrlje .....	s1521	\cyrshch .....	s1527
\CYRM .....	s1521	\CYRSHHA .....	s1527
\cym .....	s1521	\cyrshha .....	s1527
\CYRMDSC .....	s1521	\CYRT .....	s1528
\cymdsc .....	s1521	\cyrt .....	s1528
\CYRMHK .....	s1521	\CYRTDSC .....	s1528
\cymhk .....	s1521	\cyrtdesc .....	s1528
\CYRN .....	s1522	\CYRTETSE .....	s1528
\cyn .....	s1522	\cyrtetse .....	s1528
\CYRNDSC .....	s1522	\CYRTSHE .....	s1528
\cynndsc .....	s1522	\cyrtshe .....	s1528
\CYRNG .....	s1522	\CYRU .....	s1529
\cynng .....	s1522	\cyr .....	s1529
\CYRNHK .....	s1522	\CYRUSHRT .....	s1529
\cymhk .....	s1522	\cyrushrt .....	s1529
\CYRNJE .....	s1523	\CYRV .....	s1529
\cynje .....	s1522	\cyrv .....	s1529
\CYRNLHK .....	s1523	\CYRW .....	s1529
\cynlhk .....	s1523	\cyrw .....	s1529
\CYRO .....	s1523	\CYRY .....	s1529
\cyro .....	s1523	\cyry .....	s1529
\CYROTLD .....	s1523	\CYRYA .....	s1530
\cyrotld .....	s1523	\cyrya .....	s1530
\CYRP .....	s1523	\CYRYAT .....	s1530
\cyp .....	s1523	\cyryat .....	s1530
\CYRPHK .....	s1524	\CYRYHCRS .....	s1530
\cypphk .....	s1524	\crysacrs .....	s1530
\CYRQ .....	s1524	\CYRYI .....	s1530
\cyrq .....	s1524	\ciryi .....	s1530
\CYRR .....	s1524	\CYRYO .....	s1531
\cyy .....	s1524	\cyyo .....	s1530
\CYRDSC .....	s1524	\CYRYU .....	s1531
\cyyrdsc .....	s1524	\cyyu .....	s1531
\CYRRHK .....	1037, s1525	\CYRZ .....	s1531

**File Key:** a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\cyrz ..... s1531 \DeclareEmphSequence ..... 534, A541, A577, A578, A590  
 \CYRZDSC ..... s1531 \DeclareEncoding ..... 1001  
 \cyrzdsc ..... s1531 \DeclareEncodingSubset .....  
 \CYRZH ..... s1531 ..... E40, E397, E398,  
 \cyrzh ..... s1531 E399, E400, E401, E402, E403,  
 \CYRZHDSC ..... s1532 E404, E405, E406, E407, E408,  
 \cyrzhdsc ..... s1532 E409, E410, E411, E412, E413,  
**D**  
 \d ..... 1012, 1021, s235, s395, E414, E415, E416, E417, E418,  
 s482, s765, s1249, s1463, s1464, E419, E420, E421, E422, E423,  
 s1465, s1466, s1469, s1470, s1471, E424, E425, E427, E428, E429,  
 s1472, s1473, s1474, s1475, s1476, E430, E431, E432, E433, E434,  
 s1477, s1478, s1479, s1480, s1481, E435, E436, E437, E438, E439,  
 s1482, s1483, s1484, s1485, s1486, E440, E441, E442, E443, E444,  
 s1487, s1488, s1489, s1490, s1491, E445, E446, E447, E448, E449,  
 s1492, s1493, s1494, s1495, s1496, E450, E451, E452, E453, E454,  
 s1497, s1498, s1499, s1500, s1501, s1502 E455, E456, E457, E458, E459,  
 \dag ..... 1017, s312 E460, E461, E462, E463, E464,  
 \dagger ..... s312, t165, t171, t179, t180, B375 E465, E466, E467, E468, E469,  
 \dashbox ..... M309, M808, M825 E470, E471, E472, E473, E474,  
 \dashv ..... B403 E475, E476, E477, E478, E479,  
 \date ..... 743 E480, E481, E482, E483, E484,  
 \date ..... O9, O25 E485, E486, E487, E488, E489,  
 \day ..... a188, c11, T1183, T1316, T1405 E490, E491, E492, E493, E494,  
 \dblfigrule ..... 1008, X767, X2359 E495, E496, E497, E498, E499,  
 \dblfloatpagefraction P287, P301, X2336 E500, E501, E502, E503, E504,  
 \dblfloatsep ..... X753, X765, X1640, X1766, X2343 E505, E506, E507, E508, E509,  
 \dbltextfloatsep ..... X222, E510, E511, E512, E513, E514,  
 X230, X769, X1639, X1765, X2343 E515, E516, E517, E518, E519,  
 \dbltopfraction ..... P284, P298, X2335 E520, E521, E522, E523, E524,  
 \ddag ..... 1017, s313 E525, E526, E527, E528, E529,  
 \ddagger ..... s313, t166, t172, t179, t181, B374 E530, E531, E532, E533, E534,  
 \ddot ..... B518 E535, E536, E537, E538, E539,  
 \ddots ..... B513 E540, E541, E542, E543, E544,  
 \deadcycles ..... 1033, E545, E546, E547, E548, E549,  
 r334, r375, H32, H99, H165, X299 E550, E551, E552, E553, E554,  
 debug commands: E555, E556, E557, E558, E559,  
 \debug\_resume: ..... 206 E560, E561, E562, E563, E564,  
 \debug\_suspend: ..... 206 E565, E566, E567, E568, E569,  
 \DebugHooksOff ..... 191, h1444, h1550 E570, E571, E572, E573, E574,  
 \DebugHooksOn ..... 191, h1444, h1549 E575, E620, E820, E821, E822,  
 \DebugShipoutsOff ..... 876, W403, W435 E823, E865, E872, E873, E874,  
 \DebugShipoutsOn ..... 876, W403, W434 E875, E1151, E1152, E1153, E1154,  
 \DeclareAccent ..... 1001 E1155, E1156, E1157, E1158,  
 \DeclareCommandCopy ..... 94, 1051, f460, f461, f463 E1159, E1160, E1161, E1162,  
 \DeclareCurrentRelease E812, E815, T1640 E1163, E1164, E1165, E1166,  
 \DeclareDefaultHookRule 189, h1448, h1553 E1167, E1168, E1169, E1170,  
 \DeclareDocumentCommand ..... g2505 E1171, E1172, E1173, E1174,  
 \DeclareDocumentEnvironment ..... g2541 E1175, E1176, E1177, E1178,  
 \DeclareEmphSequence ..... 533 E1179, E1180, E1181, E1182,  
 File Key: a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntr1.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

E1199, E1200, E1201, E1202,  
 E1203, E1204, E1205, E1206,  
 E1207, E1208, E1209, E1210, E1211  
`\DeclareErrorFont` v477, z342, A735, B49  
`\DeclareExpandableDocumentCommand` g2561  
`\DeclareFixedFont` ..... v75  
`\DeclareFontEncoding` .... 1001, 1012,  
 s377, s463, s716, s738, s744, s830,  
 s1038, v118, B126, B127, B128, B129  
`\DeclareFontEncodingDefaults` .....  
 ..... v168, y90, y91, B36  
`\DeclareFontFamily` ... 423, v93, y85, y86  
`\DeclareFontFamilySubstitution` .. v437  
`\DeclareFontSeriesChangeRule` .....  
 ..... 434, w3, w4, w6,  
 w7, w8, w9, w10, w11, w12, w13,  
 w14, w15, w16, w17, w18, w19,  
 w20, w21, w22, w23, w24, w25,  
 w26, w27, w28, w29, w30, w31,  
 w32, w33, w34, w35, w36, w37,  
 w38, w39, w40, w41, w42, w43,  
 w44, w45, w46, w47, w48, w49,  
 w50, w51, w52, w53, w54, w55,  
 w56, w57, w58, w59, w60, w61,  
 w62, w63, w64, w65, w66, w67,  
 w68, w69, w70, w71, w72, w73,  
 w74, w75, w76, w77, w78, w79,  
 w80, w81, w82, w83, w84, w85,  
 w86, w87, w88, w89, w90, w91,  
 w92, w93, w94, w95, w96, w97,  
 w98, w99, w100, w101, w102, w103,  
 w104, w105, w106, w107, w108,  
 w109, w110, w111, w112, w113,  
 w114, w115, w116, w117, w118,  
 w119, w120, w121, w122, w123,  
 w124, w125, w126, w127, w128,  
 w129, w130, w131, w132, w133,  
 w134, w135, w136, w137, w138,  
 w139, w140, w141, w142, w143,  
 w144, w145, w146, w147, w148,  
 w149, w150, w151, w152, w153,  
 w154, w155, w156, w157, w158,  
 w159, w160, w161, w162, w163,  
 w164, w165, w166, w167, w168,  
 w169, w170, w171, w172, w173,  
 w174, w175, w176, w177, w178,  
 w179, w180, w181, w182, w183,  
 w184, w185, w186, w187, w188,  
 w189, w190, w191, w192, w193,  
 w194, w195, w196, w197, w198,  
 w199, w200, w201, w202, w203,  
 w204, w205, w206, w207, w208,  
 w209, w210, w211, w212, w213,  
 w214, w215, w216, w217, w218,  
 w219, w220, w221, w222, w223,  
 w224, w225, w226, w227, w228,  
 w229, w230, w231, w232, w233,  
 w234, w235, w236, w237, w238,  
 w239, w240, w241, w242, w243,  
 w244, w245, w246, w247, w248,  
 w249, w250, w251, w252, w253,  
 w254, w255, w256, w257, w258,  
 w259, w260, w261, w262, w263,  
 w264, w265, w266, w267, w268,  
 w269, w270, w271, w272, w273,  
 w274, w275, w276, w277, w278,  
 w279, w280, w281, w282, w283,  
 w284, w285, w286, w287, w288,  
 w289, w290, w291, w292, w293,  
 w294, w295, w296, w297, w298,  
 w299, w300, w301, w302, w303,  
 w304, w305, w306, w307, w308,  
 w309, w310, w311, w312, w313,  
 w314, w315, w316, w317, w318,  
 w319, w320, w321, w322, w323,  
 w324, w325, w326, w327, w328,  
 w329, w330, w331, w332, w333,  
 w334, w335, w336, w337, w338,  
 w339, w340, w341, w342, w343,  
 w344, w345, w346, w347, w348,  
 w349, w350, w351, w352, w353,  
 w354, w355, w356, w357, w358,  
 w359, w360, w361, w362, w363,  
 w364, w365, w366, w367, w368,  
 w369, w370, w371, w372, w373,  
 w374, w375, w376, w377, w378,  
 w379, w380, w381, w382, w386, w388  
`\DeclareFontSeriesDefault` ..... 516  
`\DeclareFontSeriesDefault` ..... 516,  
 519, 522, 523, 527, 531, A34, A35,  
 A49, A71, A72, A82, A93, A102, A104  
`\DeclareFontShape` ..... 995, v18,  
 v446, v447, v448, v449, v450, v451,  
 v452, v453, v454, v455, v456, v457,  
 v458, v459, v460, v461, v462, v463,  
 v464, v465, v466, y25, y27, y81, y82  
`\DeclareFontShapeChangeRule` .....  
 .. w523, w524, w541, w542, w543,  
 w544, w545, w546, w547, w548,  
 w549, w550, w551, w552, w553,  
 w554, w555, w556, w557, w558,  
 w559, w560, w561, w562, w563,  
 w564, w565, w566, w567, w568,  
 w569, w570, w571, w572, w573,  
 w574, w575, w576, w577, w578,  
 w579, w580, w581, w582, w583,  
 w584, w585, w586, w587, w588,  
 w589, w590, w591, w592, w593

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltphphen.dtx, Z=ltfinal.dtx

w594, w595, w596, w597, w601, w603  
 \DeclareFontSubstitution .....  
 .... s745, s831, v141, B26, B34,  
 B37, B38, B130, B131, B132, B133  
 \DeclareHookRule ..... 181, 184,  
 188, 189, 191, 193, h1446, h1552, H40  
 \DeclareHookrule ..... 188  
 \DeclareKeys ..... 841, U123  
 \DeclareMathAccent . z696, B516, B517,  
 B518, B519, B520, B521, B522,  
 B523, B524, B525, B526, B527, B528  
 \DeclareMathAlphabet .....  
 .... y119, y123, y125, y132,  
 z522, z685, B151, B152, B153, B154  
 \DeclareMathAlphabetCharacter ... z871  
 \DeclareMathDelimiter .... 560, 1033,  
 1035, z873, B255, B256, B257,  
 B258, B259, B260, B263, B265,  
 B266, B563, B565, B567, B569,  
 B571, B574, B576, B578, B580,  
 B582, B584, B586, B588, B590,  
 B592, B594, B596, B598, B600,  
 B602, B604, B606, B608, B610, B612  
 \DeclareMathOperator ..... 1017, 1021  
 \DeclareMathRadical ..... z1008, B529  
 \DeclareMathSizes .....  
 .... 1019, v205, v211, v233, B157,  
 B158, B159, B160, B161, B162,  
 B163, B164, B165, B166, B167, B168  
 \DeclareMathSizes\* ..... v205  
 \DeclareMathSymbol .....  
 .... 556, 1033, z809, z872,  
 z889, B169, B170, B171, B172,  
 B173, B174, B175, B176, B177,  
 B178, B179, B180, B181, B182,  
 B183, B184, B185, B186, B187,  
 B188, B189, B190, B191, B192,  
 B193, B194, B195, B196, B197,  
 B198, B199, B200, B201, B202,  
 B203, B204, B205, B206, B207,  
 B208, B209, B210, B211, B212,  
 B213, B214, B215, B216, B217,  
 B218, B219, B220, B221, B222,  
 B223, B224, B225, B226, B227,  
 B228, B229, B230, B231, B232,  
 B233, B234, B235, B236, B237,  
 B238, B239, B240, B241, B242,  
 B243, B244, B245, B246, B247,  
 B248, B249, B250, B251, B261,  
 B262, B264, B268, B269, B270,  
 B271, B272, B273, B274, B275,  
 B276, B277, B278, B279, B280,  
 B281, B282, B283, B284, B285,  
 B286, B287, B288, B289, B290,  
 B291, B292, B293, B294, B295,  
 B296, B297, B298, B299, B300,  
 B301, B302, B303, B304, B305,  
 B306, B307, B308, B309, B310,  
 B311, B312, B313, B314, B315,  
 B316, B317, B318, B319, B320,  
 B321, B322, B323, B324, B325,  
 B327, B328, B329, B330, B331,  
 B332, B333, B334, B341, B342,  
 B343, B344, B345, B346, B347,  
 B349, B350, B351, B352, B353,  
 B354, B356, B357, B358, B359,  
 B360, B361, B364, B365, B366,  
 B367, B370, B371, B372, B373,  
 B374, B375, B376, B377, B378,  
 B379, B380, B381, B382, B383,  
 B384, B385, B386, B387, B388,  
 B389, B390, B391, B392, B393,  
 B394, B395, B396, B397, B398,  
 B399, B400, B401, B402, B403,  
 B404, B405, B406, B407, B408,  
 B409, B410, B411, B414, B415,  
 B418, B419, B420, B421, B422,  
 B423, B424, B425, B426, B427,  
 B428, B429, B430, B432, B433,  
 B434, B435, B436, B437, B438,  
 B441, B442, B443, B445, B446,  
 B447, B448, B449, B450, B451,  
 B452, B453, B454, B455, B477,  
 B479, B501, B502, B503, B553,  
 B554, B555, B556, B614, B615, B616  
 \DeclareMathVersion ..... z355, A2, A3  
 \DeclareOldFontCommand .... D125, D141  
 \DeclareOption ..... 794  
 \DeclareOption . 1023, s1535, x29, x37,  
 x45, x53, x56, x60, E820, E821,  
 E822, E823, E824, E826, E828,  
 E830, E831, E872, E873, E874,  
 E875, E876, E878, E879, T433, T1082  
 \DeclareOption\* ..... 794  
 \DeclareOption\* ..... T433  
 \DeclarePreloadSizes .....  
 .... v185, y95, y96, C19, C21,  
 C22, C23, C25, C26, C27, C28,  
 C29, C30, C34, C38, C43, C45,  
 C49, C50, C53, C54, C57, C58, C64  
 \DeclareProtectedCommand .... 1009–1011  
 \DeclareRelease ..... E811, E814, T1547  
 DeclareRelease commands:  
 \DeclareRelease: ..... T1550  
 \DeclareRobustCommand .....  
 .... 90, 91, 94, 97, 98, 100, 101,  
 366, 453, 623, 1010–1012, 1052,  
 f221, f728, f729, f735, f750, 14, 111,

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

\DeclareSymbolFont . . . . .  
 . y136, z403, B141, B142, B143, B144  
\DeclareSymbolFontAlphabet . . . . .  
. . . . . z1082, B148, B149, B150  
\DeclareTextAccent . . . . . s64,  
s378, s379, s380, s381, s382, s383,  
s384, s385, s386, s387, s388, s464,  
s465, s466, s467, s468, s469, s470,  
s471, s472, s473, s474, s741, s746,  
s747, s748, s749, s750, s751, s752,  
s753, s754, s755, s756, s838, s839,  
s840, s841, s842, s843, s844, s845,  
s846, s847, s848, s849, s850, s851, s852  
\DeclareTextAccentDefault . . . . .  
. . . . . 1016, s185, s226, s227,  
s228, s229, s230, s231, s232, s233,  
s234, s235, s236, s237, s238, s239,  
s279, s282, E687, E688, E940, E941,  
E942, E943, E944, E945, E946,  
E947, E948, E949, E950, E951, E952  
\DeclareTextCommand 1010, 1012, 1017,  
s3, s58, s65, s83, s389, s392, s395,  
s409, s410, s411, s414, s415, s422,  
s424, s426, s428, s434, s436, s438,  
s445, s475, s478, s482, s485, s487,  
s490, s492, s494, s510, s568, s569,  
s570, s730, s757, s759, s762, s765,  
s798, s805, s832, s835, s865, s893,  
s1053, s1080, E376, E377, E378,  
E379, E380, E381, E382, E383,  
E384, E385, E625, E632, E639, E759  
\DeclareTextCommandDefault 1016, s57,  
s186, s188, s283, s286, s287, s288,  
s290, s292, s296, s300, s301, s303,  
s304, s305, s306, s326, s355, E155,  
E157, E160, E162, E164, E166,  
E168, E170, E172, E174, E176,  
E178, E180, E182, E184, E186,  
E188, E190, E193, E194, E195,  
E196, E197, E198, E199, E200,  
E201, E202, E203, E204, E205,  
E206, E207, E208, E210, E212,  
E214, E216, E218, E220, E222,  
E224, E226, E228, E230, E232,  
E234, E236, E238, E240, E242,  
E244, E246, E248, E250, E252,  
E254, E256, E258, E260, E262,  
E264, E266, E268, E270, E272,  
E274, E276, E278, E280, E282,  
E284, E286, E288, E290, E292,  
E294, E296, E298, E300, E302,  
E304, E306, E309, E311, E313,  
E315, E317, E319, E321, E323,  
E325, E327, E329, E331, E333,

**File Key:** a=ltdirname.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltppageno.dtx, G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

E335, E337, E339, E341, E343,  
 E345, E347, E349, E351, E353,  
 E355, E357, E359, E361, E363,  
 E365, E654, E662, E664, E670,  
 E678, E1006, E1008, E1009, E1011,  
 E1013, E1015, E1017, E1019,  
 E1021, E1023, E1025, E1027,  
 E1029, E1031, E1033, E1035,  
 E1037, E1039, E1041, E1043,  
 E1045, E1047, E1049, E1051,  
 E1053, E1055, E1057, E1059,  
 E1061, E1063, E1065, E1067,  
 E1069, E1071, E1073, E1075,  
 E1077, E1079, E1081, E1083,  
 E1085, E1087, E1089, E1091,  
 E1093, E1095, E1097, E1099,  
 E1101, E1103, E1105, E1107,  
 E1109, E1111, E1113, E1115,  
 E1117, E1119, E1121, E1123, E1126  
`\DeclareTextComposite` .....  
     ..... 1025, s76, s452, s453,  
 s587, s588, s589, s590, s591, s592,  
 s593, s594, s595, s596, s597, s598,  
 s599, s600, s601, s602, s603, s604,  
 s605, s606, s607, s608, s609, s610,  
 s611, s612, s613, s614, s615, s616,  
 s617, s618, s619, s620, s621, s622,  
 s623, s624, s625, s626, s627, s628,  
 s629, s630, s631, s632, s633, s634,  
 s635, s636, s637, s638, s639, s640,  
 s641, s642, s643, s644, s645, s646,  
 s647, s648, s649, s650, s651, s652,  
 s653, s654, s655, s656, s657, s658,  
 s659, s660, s661, s662, s663, s664,  
 s665, s666, s667, s668, s669, s670,  
 s671, s672, s673, s674, s675, s676,  
 s677, s678, s679, s680, s681, s682,  
 s683, s684, s685, s686, s687, s688,  
 s689, s690, s691, s692, s693, s694,  
 s695, s696, s697, s812, s813, s814,  
 s815, s816, s817, s818, s819, s820,  
 s821, s822, s823, s824, s825, s826, s827  
`\DeclareTextCompositeCommand` .....  
     ..... 1017, s76, s431,  
 s454, s455, s456, s457, s459, s698,  
 s699, s701, s704, s705, s706, s707,  
 s708, s709, s710, s711, s712, s795, s1059  
`\DeclareTextFontCommand` .....  
     ..... 1034, D1, D15, D16,  
 D17, D18, D19, D20, D21, D22,  
 D23, D24, D29, D30, D31, D42, D140  
`\DeclareTextFoo` ..... 1001  
`\DeclareTextGlyph` ..... 1008  
`\DeclareTextSymbol` ..... 1016,  
  
`\DeclareTextSymbolDefault` .....  
     ..... 1016, s185, s240, s241,  
 s242, s243, s244, s245, s246, s247,  
 s248, s249, s250, s251, s252, s253,  
 s254, s255, s256, s257, s258, s259,  
 s260, s261, s262, s263, s264, s265,  
 s266, s267, s268, s269, s270, s271,  
 s272, s273, s274, s275, s276, s277,  
 s278, s280, s281, s291, E114, E116,  
 E118, E120, E121, E122, E123,  
 E124, E125, E126, E127, E128,

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=lt hyphen.dtx, Z=ltfinal.dtx

E129, E130, E131, E132, E133,  
 E134, E135, E136, E137, E138,  
 E139, E140, E141, E142, E143,  
 E144, E145, E146, E147, E148,  
 E149, E150, E151, E152, E153,  
 E154, E1577, E1578, E1579, E1580,  
 E1581, E1582, E1583, E1584, E1597,  
 E1598, E1599, E1600, E1601, E1602,  
 E1603, E1604, E1623, E1624, E1642,  
 E1643, E1644, E1645, E1646, E1647,  
 E1648, E1650, E1953, E1954, E1955,  
 E1956, E1957, E1958, E1959, E1960,  
 E1961, E1962, E1963, E1964, E1965,  
 E1966, E1967, E1968, E1969, E1970,  
 E1971, E1972, E1973, E1974, E1975,  
 E1976, E1977, E1978, E1979, E1980,  
 E1981, E1982, E1983, E1984, E1985,  
 E1986, E1987, E1988, E1989, E1990,  
 E1991, E1992, E1993, E1994, E1995,  
 E1996, E1997, E1998, E1999, E1000,  
 E1001, E1002, E1003, E1004, E1005  
 \DeclareUnicode... . . . . . 385  
 \DeclareUnicodeAccent 385, 1045, 1059,  
 s1042, s1237, s1238, s1239, s1240,  
 s1241, s1242, s1243, s1244, s1245,  
 s1246, s1247, s1248, s1249, s1250, s1251  
 \DeclareUnicodeCharacter . . . . . 1046, Z370  
 \DeclareUnicodeCommand . . . . .  
     . . . . . 1059, s1080, s1081, s1082,  
 s1083, s1164, s1166, s1168, s1215, s1252  
 \DeclareUnicodeComposite . . . . .  
     . . . . . 385, s1057, s1256,  
 s1257, s1258, s1259, s1260, s1261,  
 s1262, s1263, s1264, s1265, s1266,  
 s1267, s1268, s1269, s1270, s1271,  
 s1272, s1273, s1274, s1275, s1276,  
 s1277, s1278, s1279, s1280, s1281,  
 s1282, s1283, s1284, s1285, s1286,  
 s1287, s1288, s1289, s1290, s1291,  
 s1292, s1293, s1294, s1295, s1296,  
 s1297, s1298, s1299, s1300, s1301,  
 s1302, s1303, s1304, s1305, s1306,  
 s1307, s1308, s1309, s1310, s1311,  
 s1312, s1313, s1314, s1315, s1316,  
 s1317, s1318, s1319, s1320, s1321,  
 s1322, s1323, s1324, s1325, s1326,  
 s1327, s1328, s1329, s1330, s1331,  
 s1332, s1333, s1334, s1335, s1336,  
 s1337, s1338, s1339, s1340, s1341,  
 s1342, s1343, s1344, s1345, s1346,  
 s1347, s1348, s1349, s1350, s1351,  
 s1352, s1353, s1354, s1355, s1356,  
 s1357, s1358, s1359, s1360, s1361,  
 s1362, s1363, s1364, s1365, s1366,  
                   s1367, s1368, s1369, s1370, s1371,  
                   s1372, s1373, s1374, s1375, s1376,  
                   s1377, s1378, s1379, s1380, s1381,  
                   s1382, s1383, s1384, s1385, s1386,  
                   s1387, s1388, s1389, s1390, s1391,  
                   s1392, s1393, s1394, s1395, s1396,  
                   s1397, s1398, s1399, s1400, s1401,  
                   s1402, s1403, s1404, s1405, s1406,  
                   s1407, s1408, s1409, s1410, s1411,  
                   s1412, s1413, s1414, s1415, s1416,  
                   s1417, s1418, s1419, s1420, s1421,  
                   s1422, s1423, s1424, s1425, s1426,  
                   s1427, s1428, s1429, s1430, s1431,  
                   s1432, s1433, s1434, s1435, s1436,  
                   s1437, s1438, s1439, s1440, s1441,  
                   s1442, s1443, s1444, s1445, s1446,  
                   s1447, s1448, s1449, s1450, s1451,  
                   s1452, s1453, s1454, s1455, s1456,  
                   s1457, s1458, s1459, s1460, s1461,  
                   s1462, s1463, s1464, s1465, s1466,  
                   s1467, s1468, s1469, s1470, s1471,  
                   s1472, s1473, s1474, s1475, s1476,  
                   s1477, s1478, s1479, s1480, s1481,  
                   s1482, s1483, s1484, s1485, s1486,  
                   s1487, s1488, s1489, s1490, s1491,  
                   s1492, s1493, s1494, s1495, s1496,  
                   s1497, s1498, s1499, s1500, s1501, s1502  
 \DeclareUnicodeSymbol . . . . .  
     . . . . . 1059, s1079, s1084, s1085,  
 s1086, s1087, s1088, s1089, s1090,  
 s1091, s1092, s1093, s1094, s1095,  
 s1096, s1097, s1098, s1099, s1100,  
 s1101, s1102, s1103, s1104, s1106,  
 s1107, s1108, s1109, s1110, s1111,  
 s1112, s1113, s1114, s1115, s1116,  
 s1117, s1118, s1119, s1120, s1122,  
 s1123, s1124, s1125, s1126, s1127,  
 s1128, s1129, s1130, s1131, s1132,  
 s1133, s1134, s1135, s1136, s1137,  
 s1138, s1139, s1140, s1141, s1142,  
 s1143, s1144, s1145, s1146, s1147,  
 s1148, s1149, s1150, s1151, s1152,  
 s1153, s1154, s1155, s1156, s1157,  
 s1158, s1159, s1160, s1161, s1162,  
 s1163, s1170, s1171, s1172, s1173,  
 s1174, s1175, s1176, s1177, s1178,  
 s1179, s1180, s1181, s1182, s1183,  
 s1184, s1185, s1186, s1187, s1188,  
 s1189, s1190, s1191, s1192, s1193,  
 s1194, s1195, s1196, s1197, s1198,  
 s1199, s1200, s1201, s1202, s1203,  
 s1204, s1205, s1206, s1207, s1208,  
 s1209, s1210, s1211, s1213, s1214,  
 s1226, s1227, s1228, s1229, s1230,

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

s1231, s1232, s1233, s1234, s1235, s1236  
 \DeclareUnknownKeyHandler 842, 1060, U125  
 \DeclareUnknownKeysHandler ..... 1060  
 \def .. 30, 251, 408, 517, 524, 548, 659,  
       992, 994, 997, 1011, 1019, 1023, 1028  
 \defaultencoding ..... 1013  
 \defaulthyphenchar ..... b367, f738, f753  
 \defaultscriptratio ..... 1006, v624, v631  
 \defaultscriptscriptratio .....  
       1006, v625, v631  
 \defaultskewchar ..... b368  
 \deg ..... I34  
 \delcode ..... z1006  
 \delimiter ..... 1044, z919, z989, z1000  
 \delimiterfactor ..... b369  
 \delimitershortfall ..... b379  
 \Delta ..... B298  
 \delta ..... B271  
 \depth ..... K32, K35  
 \det ..... I30  
 \detokenize ..... 104,  
       339, 858, c93, c94, c179, f527, f545,  
       f609, l249, l250, r253, s998, s1040,  
       w434, A563, E44, H130, T815,  
       T1154, T1158, T1286, T1287, T1291  
 \DH ..... 1012, s528, s1128, Z569  
 \dh ..... 1012, s538, s1134, Z569  
 \Diamond ..... A727  
 \diamond ..... B380  
 \diamondsuit ..... B332  
 \dim ..... I28  
 dim commands:  
     \dim\_eval:n ..... e161  
     \dim\_new:N . W197, W198, W199, W200  
     \dim\_set:Nn W191, W192, W193, W194  
     \dim\_use:N ..... W500, W501, W502  
     \c\_max\_dim . W211, W237, W262, W289  
     \c\_zero\_dim ..... n60, W217,  
       W218, W219, W225, W243, W244,  
       W245, W268, W269, W270, W297,  
       W298, W299, W327, W329, W330  
 \dimen ..... 1031  
 \dimendef .. b42, b43, b44, b52, b91, d226  
 \dimenzero ..... d226  
 \dimeval ..... 76  
 \dimeval ..... 76, 1059, e155, e171  
 \dimexpr ..... s503,  
       s519, s869, s872, s1219, s1222, M13  
 \directlua a12, a15, a20, a23, a28, b65,  
       b81, b105, b256, b349, b359, d2,  
       d14, d29, d213, d231, d256, d261,  
       d265, f34, f649, f734, s1011, T1139,  
       T1270, Z84, Z86, Z94, Z99, Z101  
 disable commands:  
     \disable\_callback ..... d814  
     \disabled\_callback ..... 48  
     \DisableGenericHook 182, 209, h1402, h1536  
     \DisableHook ..... 182, 256, h1487  
     \DiscardShipoutBox ..... 185, 872,  
       874, 877, W82, W400, W433, W489  
     \discretionary ..... f736, f751, f761, I238  
     \displaylines ..... II90  
 displaymath (env.) ..... I332  
 \displaymath ..... I334  
 \displaystyle ..... B532,  
       B535, B539, B541, I62, I197, I359,  
       I362, I415, I440, I452, I480, I505, I508  
 \displaywidowpenalty ..... b327  
 \displaywidth ..... I197, I358, I427, I483  
 \div ..... B383  
 \DJ ..... 1012, s529, s1138, Z569  
 \dj ..... 1012, s539, s1139, Z569  
 \do ..... 1026, a77, a78,  
       a129, b13, b14, f69, g1710, k3, k7,  
       k16, k26, r66, r69, r134, r137, r189,  
       r192, r232, r304, r359, r498, r515,  
       r658, r664, D90, H431, H452, H560,  
       H570, H576, H582, K57, K76, L244,  
       L269, M104, M116, M123, M203,  
       M337, M339, M362, M365, M394,  
       M396, M417, M422, M478, M506,  
       M535, M731, M787, P65, P134,  
       R36, R65, R82, T230, T247, T478,  
       T497, T514, T532, T537, T551,  
       T556, T592, T602, T1071, T1108,  
       T1186, T1239, T1319, T1408, T1685  
 \DocInput ..... x8, B5, C5, Y4  
 \docclearpage ..... 1038  
 \document ..... 330  
 document (env.) ..... H8  
 \document .....  
       72, 201, 454, 519, 523, 622, 1006,  
       1028, 1033, 1039, r9, H183, R64, R81  
 \documentclass ..... 186, 187, 308,  
       792, d16, x2, B2, C2, T609, T616,  
       T675, T678, T885, T980, T1091, Y2  
 \DocumentMetadata .....  
       308, 309, o5, o10, o14, o16, o22  
 \documentstyle ..... T614, T1091  
 \dollar ..... 1013  
 \dospecials ..... 1030,  
       a77, a129, b13, g1711, H431, H452,  
       H560, H570, T1186, T1319, T1408  
 \dot ..... B525  
 \doteq ..... B469  
 \dotfill ..... b513, f776, f797  
 \dots ..... 1017, 1020, s320, s322

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\doublehyphendemerits ..... b330  
 \doublerulesep ..... L311, L338, L362  
 \Downarrow ..... B586  
 \downarrow ..... B580  
 \downbracefill ..... B538, B557  
 \dump ..... Z635

**E**

\E ..... T1193, T1196, T1224, T1326, T1329, T1356, T1415, T1418, T1446  
 \edef ..... 30, 76, 446, 447, 495, 527, 992, 996, 1022, 1051  
 \egroup ..... 993, 1005, 1007, b417  
 \eject ..... b480  
 \ell ..... B311  
 \else ..... 1004  
 else commands:  
     \else: ..... g1155, h448, h691, h698, h706, h848, h1148, h1219, h1264, h1294, n29, V241  
 \em ..... 534, 535, A550, A577, D42  
 \emergencystretch .. 1002, 1022, S59, S65  
 \emforce ..... 533  
 \emforce .... 534, 535, A546, A573, A582  
 \eminnershape 533, 534, A554, A560, A577  
 \emph ..... 534, 1041, 1049, h1423, D42  
 \empty ..... b415  
 \emptyset ..... B318  
 \emreset ..... 534  
 \emreset .... 534, A546, A549, A580  
 \emrest ..... A549  
 \ENC-cmd ..... 1012  
 \encodingdefault 523, 1013, d260, d275, d283, s990, s1536, s1569, s1570, z347, A658, A676, A692, A709, B50  
 \EncodingSpecific .... 1001, 1002, 1008  
 \EncodingSpecificAccent ..... 1001  
 \EncodingSpecificAccentedLetter .. 1001  
 \EncodingSpecificCommand ..... 1001  
 \end .. 141, 146, 181, 199–201, 203, 224, 251, 287, 622–624, 660, 1034, 1055, a72, f23, f671, g1203, g1319, g1334, l250, x9, B6, C6, H190, H197, H199, H227, H253, H410, H411, J112, O15, O17, T1201, T1205, T1212, T1334, T1338, T1344, T1423, T1427, T1433, W385, Y5, I461, I470  
 \endarray ..... L171  
 \endcenter ..... H352  
 \endcsname ..... 186, 861, 863, 992, 1075  
 \enddisplaymath ..... I335  
 \enddocument ..... 198, 201, 202, 616, 618, 870, 891, 892, 1026, 1053, 1054, H8, H66, H68, W350

\endenumerate ..... J240  
 \endeqnarray ..... I367, I414  
 \endequation ..... I338  
 \endfilecontents ..... T1095  
 \endflushleft ..... H402  
 \endflushright ..... H404  
 \endgraf ..... 296, 303, 306, 1000, 1005, 1007, 1016, 1020, 1030, 1031, b412, n98, n136, K94  
 \endgroup ..... 332, 622, 863, 992, 993, 996, 1005, 1034  
 \EndIncludeInRelease .....  
     ... 30, a25, a53, a309, a319, b87, b101, b118, b123, b133, b137, b147, b150, b167, b181, b185, b219, b224, b232, b240, b288, b300, b355, b363, b438, b455, b546, b580, b588, b613, b653, b658, b668, b673, c68, d234, d257, d280, d284, e10, e18, e27, e110, e131, e144, e153, e165, e173, e188, e194, f12, f18, f60, f64, f312, f340, f368, f373, f392, f406, f447, f457, f487, f496, f510, f515, f613, f628, f660, f669, f719, f726, f748, f759, f762, f790, f811, g1158, g1161, g1304, g1308, g2630, g2637, h138, h144, h163, h169, h387, h415, h467, h495, h519, h533, h553, h556, h569, h586, h591, h594, h600, h1133, h1161, h1164, h1190, h1403, h1409, i30, i44, i388, i400, i157, i161, l204, l210, p20, p30, p64, p77, p85, p90, p103, p109, p151, p165, p177, p188, p205, p217, p251, p268, p306, p328, p365, p398, p431, p435, p444, p450, p464, p471, r82, r139, r194, r255, r276, r288, r346, r380, r418, r441, r464, r482, r489, r508, r525, r538, r542, r560, r571, r581, r620, r647, s103, s123, s168, s175, s330, s352, s367, s375, t28, t33, t68, t73, t98, t125, t134, t176, t182, t202, t205, u10, u14, v52, v73, v231, v248, v294, v304, v314, v430, v435, v470, v475, v492, v510, v549, v582, v698, v710, w384, w390, w403, w415, w422, w505, w520, w599, w611, w622, w629, w636, w709, w727, w734, w738, w741, x157, x160, x176, x549, x555, y21, y143, z77, z105, z216, z246, z276, z308, z364, z369, z745, z787, z800, z807, z995, z1003, A67, A100, A122, A227, A248, A300, A347, A376,

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

A387, A434, A468, A494, A530,  
 A539, A576, A588, A595, A629,  
 A635, A670, A686, A703, A718,  
 B80, B90, B109, B118, B633, B640,  
 D33, D40, G50, G61, G72, H64,  
 H103, H111, H117, H150, H163,  
 H225, H275, H294, H308, H317,  
 H327, H334, H344, H349, H373,  
 H389, H398, H436, H457, H488,  
 H498, H516, H531, H543, H565,  
 H573, J132, J137, K13, K22, K69,  
 K86, K101, K113, K124, K131,  
 K188, K195, K243, K251, K302,  
 K320, K388, K404, K421, K430,  
 K435, K458, K465, L64, L69, L156,  
 L164, L226, L231, M15, M19, M38,  
 M47, M68, M76, M88, M96, M111,  
 M121, M150, M155, M171, M182,  
 M249, M263, M368, M425, M463,  
 M469, M500, M530, M555, M571,  
 M582, M595, M603, M624, M641,  
 M651, M655, M745, M800, M819,  
 M836, O19, O29, O167, O173,  
 O178, O192, O198, O224, O246,  
 P104, P172, P231, P246, P293,  
 P306, P351, P368, P411, P417,  
 P426, P430, P439, P445, P449,  
 P481, P498, P516, P558, P564,  
 R24, R32, R76, R92, S39, S46, T21,  
 T26, T49, T61, T84, T93, T99,  
 T111, T142, T149, T173, T179,  
 T197, T208, T242, T259, T270,  
 T279, T296, T308, T329, T342,  
 T355, T365, T401, T417, T426,  
 T458, T468, T508, T524, T546,  
 T561, T575, T582, T597, T606,  
 T640, T647, T664, T671, T750,  
 T777, T805, T955, T1012, T1042,  
 T1049, T1227, T1358, T1448, V14,  
 V23, V88, V140, V179, V191,  
 V200, V245, V256, V263, V299,  
 V322, V343, V357, V362, V385,  
 V405, V430, V475, V491, V498,  
 V528, V533, W418, W455, W472,  
 W476, X53, X62, X178, X196,  
 X365, X370, X418, X464, X650,  
 X708, X811, X830, X893, X911,  
 X953, X974, X1216, X1385, X1467,  
 X1561, X1683, X1810, X1920,  
 X1928, X1962, X1991, X2211,  
 X2229, X2276, X2320, Z15, Z19,  
 Z37, Z55, Z65, Z72, Z80, Z127,  
 Z132, Z184, Z208, Z278, Z283,  
 Z324, Z330, Z426, Z473, I86, I95,  
 I117, I124, I143, I147, I162, I170,  
 I219, I236, I266, I274, I303, I330,  
 I394, I403, I443, I455, I464, I473  
 \EndIncludeRelease . . . . . c75  
 \endinput . . . . . 1006  
 \enditemize . . . . . J251  
 \endline . . . . . b412, I175  
 \endlinechar . . . . . 619,  
     1038, a95, a96, a97, a207, f37, f39,  
     f44, r595, r629, T370, T371, T372, Z315  
 \endlist . . . . . J98, J240, J251  
 \endlrbox . . . . . K155  
 \endmath . . . . . I333  
 \endminipage . . . . . K354  
 \EndModuleRelease . . . . . c108, c109, c149,  
     g2662, h1558, i424, n141, o24, E809  
 \endpicture . . . . . M49  
 \endscname . . . . . 992  
 \endsloppypar . . . . . S63  
 \endtabbing . . . . . L84  
 \endtabular . . . . . L171  
 \endtabular\* . . . . . L171  
 \endtrivlist . . . . . H352, H402,  
     H404, H460, J100, J101, L85, N39, I485  
 \endverbatim . . . . . H459, H486  
 \enlargethispage . . . . . 1040, X1855  
 \enlargethispage\* . . . . . X1855  
 \enskip . . . . . p473  
 \enspace . . . . . p461, p469  
 \ensuremath . . . . . 1029,  
     t178, P409, P416, P437, P444, I416  
 enumerate (env.) . . . . . J231  
 \enumerate . . . . . J231  
 environments:  
     array . . . . . L168  
     center . . . . . H351  
     displaymath . . . . . I332  
     document . . . . . H8  
     enumerate . . . . . J231  
     eqnarray . . . . . I344, I486  
     eqnarray\* . . . . . I412  
     equation . . . . . I336, I474  
     filecontents . . . . . T1095, 792  
     flushleft . . . . . H401  
     flushright . . . . . H403  
     itemize . . . . . J242  
     list . . . . . J34  
     lrbox . . . . . 666  
     math . . . . . I332  
     minipage . . . . . 667  
     picture . . . . . M21  
     sloppypar . . . . . S62  
     tabbing . . . . . L71  
     tabular . . . . . L174

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=lt page.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=lt hyphen.dtx, Z=lt final.dtx

**thebibliography** . . . . . 782  
**trivlist** . . . . . J89  
**verbatim** . . . . . H459  
**verbatim\*** . . . . . H483  
**\epsilon** . . . . . B272  
**eqnarray** (env.) . . . . . I344, I486  
**\eqnarray** . . . . . I349, I413  
**\eqnarray\*** (env.) . . . . . I412  
**\eqno** . . . . . I016, I338  
**equation** (env.) . . . . . I336, I474  
**\equation** . . . . . I337  
**\equiv** . . . . . B448  
**\errhelp** a220, c31, l39, l66, Y12, Z293, Z624  
**\errmessage** . . . . .  
    . a7, a61, a225, b164, b178, b304,  
    c32, d63, e79, e94, l47, l72, v521,  
    v556, x425, x525, y65, Y16, Z49, Z295  
**\ERROR** . . . . . 158, 159, 298,  
    g1397, g1407, g1416, g1769, g1782,  
    g1802, g1821, g1897, g1924, h834, h835  
**\errorcontextlines** . . . . .  
    . 994, 1000, b372, b536, b555,  
    b571, b586, b600, b624, b641, l212  
**\errorstopmode** 190, 193, 1025, b521, Z634  
**\escapchar** . . . . . 342  
**\escapechar** . . . . . 99, d212, f126,  
    f169, f173, f181, f275, f276, f300,  
    f437, f525, f543, r399, v367, v591,  
    x229, z58, z86, z158, z226, z256,  
    z287, z331, V281, V304, V327, V348  
**\eta** . . . . . B274  
**\etacatcode** . . . . . d862  
**\TeXversion** . . . . . a60  
**\evensidemargin** . . . . . X73, X613, X672  
**\everycr** . 997, b501, I192, I195, I358, I502  
**\everydisplay** . . . . . v345, v346, v351  
**\everyeof** . . . . . Z315  
**\everyjob** . . . . . 1042, 1046, c37, c42, c47,  
    d216, d262, d263, z351, W29, W30,  
    Z98, Z317, Z402, Z594, Z595, Z597  
**\everymath** . . . . . v344, v346, v349  
**\everypar** . . . . . 290  
**\everypar** . . . . . 292–297, 300–302,  
    306, 307, 660, 1008, 1032, 1042,  
    n32, n35, n78, n88, n101, n139, r43,  
    r112, r170, v644, v657, v704, H166,  
    H433, H455, J129, J131, J135,  
    J136, J180, J197, K292, K313,  
    L81, O48, O96, O107, O127, O136,  
    P187, X165, X192, X1151, X1317  
**\EveryShipout** . . . . . 877  
**\ExecuteOptions** x57, x70, E843, E880, T585  
**\exhyphenpenalty** . . . . . b322, b474  
**\exists** . . . . . B324  
**\exp** . . . . . I31  
**exp commands:**  
    **\exp:w** . . . . . 220, h438  
    **\exp\_after:wN** . . . . . 220, g254, g270,  
        g324, g330, g344, g1025, g1039,  
        g1041, g1072, g1121, g1151, g1181,  
        g1185, g1200, g1330, g1366, g1390,  
        g1406, g1424, g1430, g1434, g1465,  
        g1499, g1506, g1519, g1525, g1531,  
        g1537, g1543, g1549, g1555, g1561,  
        g1592, g1720, g2136, g2137, g2138,  
        g2139, g2140, g2141, g2142, g2168,  
        g2169, g2170, g2180, g2249, h49,  
        h56, h61, h218, h219, h264, h437,  
        h648, h738, h847, h954, h1066,  
        h1128, h1216, h1261, i167, i220,  
        i221, i270, z186, z190, V73, V236, V237  
**\exp\_args:Nc** . . . . . g195,  
    g957, g968, g1204, g1221, g1323,  
    g2075, g2088, h1100, i28, i41, i73, i131  
**\exp\_args:Ncc** . . . . . g130  
**\exp\_args:Ne** . . . . . g2162,  
    h404, h405, h1177, h1178, V46, V97  
**\exp\_args:Nee** . . . . . V129  
**\exp\_args:Nf** . . . . .  
    . g349, g1901, g1928, g1989,  
    g2164, g2248, i158, V32, V34, V395  
**\exp\_args:NNc** . . . . . i145, i150  
**\exp\_args:NNNo** . . . . . g330, g1633  
**\exp\_args:Nnnv** . . . . . h115  
**\exp\_args:NNo** . . . . . h830, i166  
**\exp\_args:NNV** . . . . . h804, i185  
**\exp\_args:NNx** . . . . . g1220, h751  
**\exp\_args:No** . . . . . g88, g251,  
    g341, g360, g369, g456, g806, g1157,  
    g1774, g1785, g1826, g1907, g1934,  
    g2104, g2130, h1040, h1084, i166, i185  
**\exp\_args:Nof** . . . . . g1777  
**\exp\_args:Noo** . . . . . g1130  
**\exp\_args:NV** . . . . .  
    . g190, g293, g1285, i186, U131, V53  
**\exp\_args:Nv** . . . . . h983, h990, h1064  
**\exp\_args:Nx** . . . . .  
    g2103, g2106, g2129, h270, h312, W29  
**\exp\_end:** . . . . . h438  
**\exp\_last\_unbraced:Ne** . . . . . g2182  
**\exp\_last\_unbraced:Nf** . . . . . i109  
**\exp\_last\_unbraced:NNNNo** . . . . . h221, i115  
**\exp\_last\_unbraced:NnNo** . . . . . g2001  
**\exp\_last\_unbraced:NNo** . . . . . i281  
**\exp\_not:N** . . . . . 265–267,  
    g86, g114, g115, g147, g149, g150,  
    g156, g158, g199, g211, g228, g229,  
    g230, g232, g300, g815, g825, g890,

**File Key:** a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=lttoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

g903, g927, g945, g952, g990, g997,  
 g998, g1030, g1045, g1046, g1047,  
 g1051, g1054, g1055, g1057, g1076,  
 g1079, g1080, g1081, g1089, g1136,  
 g1155, g1169, g1175, g1176, g1213,  
 g1214, g1566, g1567, g1674, g1676,  
 g1677, g1742, h229, i107, i109, i110,  
 i111, i181, i196, i203, i204, i208,  
 i212, i228, i269, i275, i293, i302,  
 i354, i355, i380, i381, n37, n38,  
 U131, U132, U133, W31, W95, W124  
**\exp\_not:n** ..... 117, 264, 265,  
 267, g109, g113, g117, g120, g124,  
 g141, g148, g160, g212, g231, g300,  
 g303, g304, g538, g672, g762, g773,  
 g775, g776, g839, g891, g904, g915,  
 g927, g944, g1029, g1031, g1045,  
 g1058, g1090, g1125, g1314, g1316,  
 g1329, g1375, g1376, g1419, g1479,  
 g1675, g1792, g1793, g1833, g1834,  
 g2016, g2025, h721, i168, i181, i184,  
 i186, i214, i273, i303, i305, i306,  
 i308, i309, U115, V211, V225, W30  
**\exp\_stop\_f:** g1241, g1779, h688, h696  
 expandable commands:  
   **\expandable\_grab\_<type>:w** ..... 139  
**\expandafter** .. 491, 623, 624, 861, 993,  
 997, 1015, 1020, 1026, 1030, 1035, 1057  
 expandafter commands:  
   **\expandafter:** ..... D88, L269  
**\ExpandArgs** ..... 77  
**\ExpandArgs** ..... e174, e190, e193  
**\expanded** ..... 214, T815, Z97  
**\ExplSyntaxOff** .. 798, e143, e163, e186,  
 g2663, h1559, i298, i425, n142, o25,  
 z214, U195, V12, V86, V138, V243,  
 V254, V266, V403, V428, V473,  
 V526, V581, W416, W463, W503, Z554  
**\ExplSyntaxOn** ..... 77,  
 e135, e158, e177, g8, h3, i3, i298,  
 n3, o2, z178, U3, V7, V29, V92,  
 V207, V251, V391, V411, V436,  
 V505, V574, W5, W461, W498, Z551  
**\extracolsep** ..... L167  
**\extraffloats** ..... b152, b189, b274  
  
**F**  
**\fam** ..... b98, d22, d26, d38, v15  
**\family** ..... 992, 993, 995  
**\familydefault** .. 522, 523, s1570, z348,  
 A444, A659, A677, A693, A710, B120  
**\fbox** ..... 666  
**\fbox** ..... 1010, K173, K186, K193

**\fboxrule** ..... K171, K207,  
 K210, K216, K218, K225, K226, Z137  
**\fboxsep** ..... K171,  
 K177, K206, K211, K221, K223, Z136  
**\fi** ..... 993, 998, 1040  
 fi commands:  
   **\fi:** ..... 267, g773,  
 g776, g838, g1155, g1243, h450,  
 h692, h700, h706, h850, h1067,  
 h1129, h1150, h1221, h1266, h1296,  
 i263, i266, i271, i278, n30, n61, V241  
**\filbreak** ..... b478  
 file commands:  
   **\g\_file\_curr\_name\_str** ..... h1437  
   **\file\_full\_name:n** ..... 857, V35  
   **\file\_parse\_full\_name\_apply:nN** ..  
 ..... V32, V47, V90, V93, V95  
   **\l\_file\_search\_path\_seq** 849, 857, 864  
 file internal commands:  
   **\\_\_file\_parse\_full\_name\_area:nw** ..  
 ..... V102, V105, V109  
   **\\_\_file\_parse\_full\_name\_auxi:nN** ..  
 ..... V97, V100  
   **\\_\_file\_parse\_full\_name\_base:nw** ..  
 ..... V108, V112, V124  
   **\\_\_file\_parse\_full\_name\_tidy:nnnN** ..  
 ..... V119, V120, V122, V127  
 file/.../after ..... 848  
 file/.../before ..... 848  
 file/after ..... 848  
 file/before ..... 848  
 filecontents (env.) ..... T1095, 792  
**\filecontents** ..... T1095  
 filehook internal commands:  
   **\\_\_filehook\_clear\_replacement\_-**  
   flag: ..... V418, V421, V517  
   **\\_\_filehook\_drop\_extension:N** ..  
 ..... V44, V49, V519  
   **\\_\_filehook\_drop\_extension\_-**  
   aux:nnn ..... V54, V57  
   **\\_\_filehook\_file\_name\_compose:nnn**  
 ..... V218, V407, V415, V416, V426  
   **\\_\_filehook\_file\_parse\_full\_-**  
   name:nN ..... 854, 864, V28, V30,  
 V30, V53, V238, V395, V413, V415  
   **\\_\_filehook\_file\_pop:** V59, V70, V523  
   **\\_\_filehook\_file\_pop\_assign:nnm**  
 ..... V59, V73, V79, V525  
   **\\_\_filehook\_file\_push:** V59, V62, V521  
   **\\_\_filehook\_file\_subst\_begin:nnm**  
 ..... 864, V413, V423, V423  
   **\\_\_filehook\_file\_subst\_cycle\_-**  
   error:NN .....  
 . 866, V458, V463, V463, V468, V470

**File Key:** a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

\\_\_filehook\_file\_subst\_loop:NN . . . . .  
     865, V432, V445, V453, V462  
 \\_\_filehook\_file\_subst\_tortoise\_-  
     hare:nn . . . . . 865,  
     866, V425, V432, V435, V437, V460  
 \\_\_filehook\_full\_name:nn . . . . .  
     V30, V34, V38  
 \\_\_filehook\_if\_file\_replaced:TF . . . . .  
     865, V418, V419, V515  
 \\_\_filehook\_if\_no\_extension:nTF . . . . .  
     V44, V44, V507  
 \g\_filehook\_input\_file\_seq 855, V59  
 \l\_filehook\_internal\_tl . . . . . V59  
 \\_\_filehook\_log\_file\_record:n . . . . .  
     V544, V544, V558, V560  
 \g\_filehook\_nesting\_level\_int . . . . .  
     V541, V546, V549, V550, V556  
 \\_\_filehook\_normalize\_file\_-  
     name:w . . . . . V407, V414, V513  
 \\_\_filehook\_resolve\_file\_subst:w . . . . .  
     V407, V410, V412, V511  
 \\_\_filehook\_set\_curr\_file:nNN . . . . .  
     V393, V509  
 \\_\_filehook\_set\_curr\_file\_-  
     assign:nnnNN . . . . . V396, V398  
 \\_\_filehook\_subst\_add:nn . . . . .  
     859, V206, V208, V208, V252  
 \\_\_filehook\_subst\_empty\_name\_-  
     chk:NN . . . . . V208, V236, V240  
 \\_\_filehook\_subst\_file\_normalize:Nn . . . . .  
     V208, V216, V218, V230, V234  
 \\_\_filehook\_subst\_remove:n . . . . .  
     V208, V222, V253  
 \filename@parse . . . . . 1, 6  
 \filesize . . . . . e72, e118  
 \fill . . . . . p454  
 \finalhyphendemerits . . . . .  
     1051, 1052, b331, H360, H364, H370  
 \finalstrut . . . . . 1019  
 \firstmark . . . . . S51, X647, X706, X2251  
 flag internal commands:  
     \flag\_{\\_\_filehook\_file\_replaced} . 865  
     \flag\_{\\_\_filehook\_file\_replaced} . V418  
 flag commands:  
     \flag\_clear:n . . . . . V422  
     \flag\_if\_raised:nTF . . . . . V420, V441  
     \flag\_new:n . . . . . V418  
     \flag\_raise:n . . . . . V442  
 \flat . . . . . B328  
 \floatingpenalty . . . . P469, P488, P506  
 \floatpagefraction . . . . P278, X2332  
 \floatsep . . . . . X726,  
     X744, X751, X2138, X2188, X2337  
 \flushbottom . . . . . S55  
 \flushleft (env.) . . . . . H401  
 \flushleft . . . . . H401  
 \flushright (env.) . . . . . H403  
 \flushright . . . . . H403  
 \fmtname . . . . . 998, c1,  
     c38, c40, c43, c45, c48, c57, T684, T688  
 \fmtversion . . . . .  
     998, c1, c19, c38, c40, c43, c45,  
     c48, c57, c102, c115, c163, d276,  
     l2, s1538, A225, L1, M1, T169,  
     T176, T701, T704, X4, Z578, Z604  
 \fnsymbol . . . . . 397  
 \fnsymbol . . . . . t141  
 \font . . . . . b506, b511,  
     f737, f740, f752, f755, s297, s298,  
     s299, s439, s446, s799, s806, s866,  
     s1003, s1004, s1060, s1165, s1167,  
     s1169, s1216, v81, v87, v89, x84,  
     A553, A586, A592, A618, C8, C9,  
     C10, D85, E6, E612, E626, E633, H454  
 \fontdimen . . . . . b506, b511,  
     s297, s298, s299, s439, s446, s799,  
     s806, A553, A586, A592, D85, E6,  
     E612, E626, E633, M126, M129, M679  
 \fontencoding . . . . .  
     416, 1010, d259, d260, d282,  
     d283, s867, s1569, v251, v286, v298,  
     v308, z347, A658, A676, A692,  
     B16, B24, B67, B74, B83, B85, E36  
 \fontfamily . . . . . 416, 443, 529, 995,  
     1010, v279, A6, A9, A12, A145,  
     A156, A482, A485, A488, A738,  
     B58, B69, B76, B87, E28, E30,  
     E32, E34, E85, E87, E89, E91, E916  
 \fontname . . . . . s1004, v89  
 \fontseries . . . . .  
     416, 443, 446, 451, 459, 524, 529,  
     538, 540, 1010, 1049, 1050, v279,  
     w393, w394, w405, w407, w417,  
     w419, A15, A18, A258, A259, A260,  
     A261, A281, A282, A283, A284,  
     A319, A320, A321, A322, A339,  
     A340, A341, A342, A355, A356,  
     A357, A358, A366, A367, A368,  
     A369, A382, A385, A476, A479, A739  
 \fontseriesforce . . 443, w398, w409, w420  
 \fontshape . . 416, 451, 459, 540, 1010,  
     1049, s449, s809, v279, w528, w533,  
     w538, w614, w615, w624, w626,  
     w631, w633, w684, w688, w691,  
     w694, w697, w700, w703, w706,  
     w713, A21, A24, A27, A30, A740, E636  
 \fontshapeforce . . w618, w627, w634, w714

**File Key:** a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

```

\fontsize 1010, q6,
 s302, s328, s360, s868, s1218, s1254,
 v79, v318, A641, A741, E104,
 E672, E933, P409, P416, P437, P444
\fontsubfuzz .. 1007, x441, x475, H42, H82
\FOO 79
\foo 263
\footins 1024, P390, P465, P484,
 P502, X314, X315, X316, X317,
 X375, X422, X482, X490, X494, X517
\footnote 778, 994, P452
\footnotemark 994, 1018, O10, P518
\footnoterule ... 1024, K360, P394, X493
\footnotesep K384, K401, K418, P451,
 P468, P477, P487, P495, P505, P513
\footnotesize
 K376, K394, K411, P466, P485, P503
\footnotetext 994, O12, P535
\footref 1055, P547
\footskip X77, X637, X696
\forall B323
fp commands:
 \fp_eval:n e159
\fpeval 76
\fppeval 76, e155, e167, e169
\frac 1000, I341
\frame K157, K233
\framebox 666
\framebox 1010, 1015, K180
\frenchspacing b398, f777,
 f798, r46, r115, r173, H459, H485, H575
\frown B451
\fussy S64
\futurelet e49,
 f677, f691, p410, p418, D83, L359, I243

G
\g_@@_\meta_{hook}_code_prop 207
\Gammama B297
\gammama B270
\gcd I33
\gdef 799, 1013, 1048
gdef commands:
 \gdef_ I256
\ge B417, B419
\Generic* 1018
\GenericError 1014, l18, l85, l111, l164, x62
\GenericInfo
 1014, c103, c105, c116, c119, c124,
 c159, c160, c165, c169, c173, c194,
 e63, l4, l104, l130, l182, x31, x34,
 x39, x75, T66, T74, T104, T107, T1611
\GenericWarning .. 1014, l11, l94, l120,
 l141, l149, l173, l195, x42, x47, x50, x78

H
\H l24, s230, s385, s469,
 s601, s609, s628, s636, s753, s1246,
 s1385, s1386, s1413, s1414, E177, E201
\halign 89,
 992, 997, b501, I127, I197, I358, I502
\hang 1023
\hangindent O139
\hat B523

```

**File Key:** a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntr1.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=lt hyphen.dtx, Z=ltfinal.dtx

\hbadness ..... b318, v647, v654, v689, v708, W236, W238, W288, W290  
 \hbar ..... B335  
 \hbox ... 303, 871, 873, 876, 995, 1022, 1039  
 hbox commands:  
   \hbox:n ..... W371  
   \hbox\_set:Nn .....  
     .. W168, W215, W241, W266, W295  
   \hbox\_set\_to\_wd:Nnn .....  
     ..... W239, W291, W327  
   \hbox\_unpack:N ..... W249, W293  
   \hbox\_unpack\_drop:N ..... W178  
   \hbox\_u\_to\_u ..... 1007  
   \headheight ..... X75, X626, X685  
   \headsep ..... X76, X635, X694  
   \heartsuit ..... B333  
   \height ..... 999, s872, s1222, K31, K34  
   \hfil ..... 652  
   \hfill ..... 995  
   \hfuzz ..... b373, v655, S60, S61, S67, S68, W235, W237, W287, W289  
   \hglue ..... b468  
   \hideoutput ..... b659  
   \hideskip ..... b309, b492  
   \hidewidth ..... b492, s327,  
     s329, s358, s362, s390, s391, s394,  
     s397, s476, s477, s481, s484, s486,  
     s489, s501, s506, s522, s760, s761,  
     s764, s767, s834, s837, s1253, s1255  
   \hline ..... 996, L358, L361  
   \holdinginserts ..... b335  
   \hom ..... I29  
 hook commands:  
   \hook\_activate\_generic:n .....  
     ..... 192, 212, h145,  
       h146, h147, h165, h167, h1400,  
       h1401, h1405, h1467, h1486, h1499  
   \hook\_debug\_off: .. 193, h7, h13, h1445  
   \hook\_debug\_on: .. 193, h7, h8, h1444  
   \hook\_disable:n ..... h1455, h1455  
   \hook\_disable\_generic:n .....  
     ..... 191, h121, h122,  
       h123, h140, h142, h1402, h1460, h1492  
   \hook\_gclear\_next\_code:n .....  
     ..... 192, h1111, h1111, h1415  
   \hook\_gput\_code:n ..... 209  
   \hook\_gput\_code:nnn ... 192, 217,  
     218, h323, h323, h376, h393, h1411  
   \hook\_gput\_next\_code:nn .... 192,  
     219, h383, h398, h1086, h1086, h1413  
   \hook\_gremove\_code:nn .....  
     ..... 193, 224, h601, h601, h1417  
   \hook\_gset\_rule:nnnn ..... 193,  
     226, h636, h636, h1447, h1449, h1452  
   \hook\_if\_empty:n ..... h1225  
   \hook\_if\_empty:nTF ..... 185, 186,  
     193, h963, h1225, h1453, W63, W162  
   \hook\_if\_empty\_p:n .....  
     193, h1008, h1225, W67, W113, W366  
   \hook\_log:n ... 193, h934, h934, h1443  
   \hook\_new:n .....  
     ..... 191, 192, 209, 211, 220, 245,  
       h68, h68, h108, h506, h1396, W152,  
       W153, W154, W155, W156, W157  
   \hook\_new\_pair:nn .....  
     ..... 191, h107, h107, h1398, n6, n7  
   \hook\_new\_reversed:n .....  
     ..... 191, h100, h100, h108, h1397  
   \g\_hook\_patch\_action\_list\_tl ...  
     ..... i6, i90, i123  
   \hook\_provide:n ..... h1455, h1462  
   \hook\_provide\_pair:nn .. h1455, h1476  
   \hook\_provide\_reversed:n h1455, h1469  
   \hook\_show:n 193, 236, h934, h939, h1442  
   \hook\_use:n .....  
     185, 186, 191, 192, 196, 208, 210,  
       214, 241, 243, h724, h1115, h1115,  
       h1117, h1134, h1136, h1440, n20,  
       n28, n56, n63, W62, W65, W71, W75  
   \hook\_use\_once:n ..... 185,  
     192, 196, 243, h1191, h1191, h1441  
 hook internal commands:  
   \g\_\_hook\_??\_code\_prop ..... h633  
   \g\_\_hook\_??\_reversed\_tl ..... h633  
   \g\_\_hook\_<hook>\_code\_prop .....  
     ..... 207, 225, 244  
   \g\_\_hook\_<hook>\_labels\_clist ... 210  
   \g\_\_hook\_<hook>\_reversed\_tl ... 207  
   \\_\\_hook\_activate\_generic:n ... h145  
   \\_\\_hook\_activate\_generic:nn ....  
     ..... h148, h149, h1484  
   \\_\\_hook\_activate\_generic\_pair:nn  
     ..... h1455, h1481, h1485, h1513  
   \\_\\_hook\_activate\_generic\_-  
     reversed:n .....  
       .. h1455, h1474, h1483, h1486, h1506  
   \g\_\_hook\_all\_seq ..... h28, h83, h713  
   \\_\\_hook\_apply\_-rule\_->:nnn ... h909  
   \\_\\_hook\_apply\_-rule\_-<:nnn ... h909  
   \\_\\_hook\_apply\_-rule\_<:nnn ... h909  
   \\_\\_hook\_apply\_-rule\_>:nnn ... h909  
   \\_\\_hook\_apply\_-rule\_x:nnn .... h909  
   \\_\\_hook\_apply\_label\_pair:nnn ...  
     ..... 232, 233, 239, h781, h782, h836, h836  
   \\_\\_hook\_apply\_rule:nnn .....  
     ..... 234, h846, h852, h852  
   \\_\\_hook\_apply\_rule:nnnN ..... 234  
   \\_\\_hook\_apply\_rule\_->:nnn .... h887

**File Key:** a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

```

__hook_apply_rule_<:nnn h887
__hook_apply_rule_<:nnn h859
__hook_apply_rule_>:nnn h859
__hook_apply_rule_xE:nnn h873
__hook_apply_rule_xW:nnn h873
__hook_clean_to_scan:w
..... h39, h39, h1216, h1261
__hook_clear_next:n
..... h1108, h1111, h1112, h1113
__hook_clist_gput:Nn
..... h745, h747, h805, h834, h835
__hook_cmd_begindocument_code: ..
..... 272, i45, i53, i58, i61, i422
__hook_cmd_if_scannable:Nn i345
__hook_cmd_if_scannable:NnTF ...
..... 269, 271, i323, i345
__hook_cmd_patch_xparse:Nnn ...
..... i127, i148, i148
__hook_cmd_try_patch:nn i51, i62, i62
\l__hook_cur_hook_tl
..... 235, h29, h767, h893, h904
__hook_curr_name_pop:
..... 216, h255, h287, h1433, h1527
__hook_curr_name_push:n ... 215,
..... 216, 250, h255, h269, h1431, h1436
__hook_curr_name_push_aux:n ...
..... h255, h270, h271
__hook_currname_or_default: ...
..... 213, h173, h181, h185,
..... h201, h202, h202, h350, h1326, h1356
__hook_debug:n .. 204, h7, h7, h20,
..... h344, h712, h717, h729, h751, h786,
..... h806, h861, h868, h875, h883, h889,
..... h900, i18, i27, i38, i47, i48, i64, i68
\g__hook_debug_bool h6, h10, h15, h21
__hook_debug_gset: h7, h11, h16, h18
__hook_debug_label_data:N ...
..... h786, h827, h922, h922
__hook_debug_print_rules:n ...
..... h1070, h1070
__hook_declare_DEPRECATED-
..... generic:NNn h437, h458
__hook_declare_DEPRECATED-
..... generic:NNw h453, h459, h460
__hook_def_cmd:w
..... 265, i12, i13, i189, i195, i220
\g__hook_delayed_patches_prop ...
..... i16, i50, i56, i57
__hook_DEPRECATED_generic-
..... warn:n
..... 221, h436, h443, h627, h645, h952
__hook_DEPRECATED_generic-
..... warn:Nn h443
__hook_DEPRECATED_generic-
..... warn:Nw h443
__hook_DEPRECATED_generic-
..... warn:w h444, h445
__hook_DEPRECATED_warn:nn
..... h1457, h1464, h1471, h1478, h1489,
..... h1496, h1503, h1510, h1515, h1515
__hook_disable:n .. h121, h124, h125
__hook_do_DEPRECATED_generic:Nn
..... h453, h453, h628, h646, h953
__hook_do_DEPRECATED_generic:Nw
..... h453, h454, h455
__hook_double_hashes:n
..... 264, 266, i185, i246, i246, i280
__hook_double_hashes:w
..... 266, i246, i247, i248, i276, i280, i283
__hook_double_hashes_group:n ...
..... i246, i254, i279
__hook_double_hashes_output:N ..
..... 267, i246, i251, i259
__hook_double_hashes_space:w ...
..... i246, i255, i282
__hook_double_hashes_stop:w ...
..... i246, i262, i278
__hook_end_document_label-
..... check: h255, h294, h295, h302
__hook_exp_not:n
..... 264, i168, i169, i173, i184
__hook_exp_not:NN
..... i12, i12, i190, i196, i212, i214
__hook_file_hook_normalize:n ...
..... 223,
..... h405, h554, h554, h557, h559, h1178
\l__hook_front_tl h758, h797, h800,
..... h803, h805, h806, h807, h820, h821
\c__hook_GENERIC_(type)/./<place>_tl
..... 220
\c__hook_GENERIC_class/./after-
..... tl h570
\c__hook_GENERIC_class/./before-
..... tl h570
\c__hook_GENERIC_cmd/./after_tl h570
\c__hook_GENERIC_cmd/./before_tl
..... h570
\c__hook_GENERIC_env/./after_tl h570
\c__hook_GENERIC_env/./before_tl
..... h570
\c__hook_GENERIC_env/./begin_tl h570
\c__hook_GENERIC_env/./end_tl . h570
\c__hook_GENERIC_file/./after_tl
..... h570
\c__hook_GENERIC_file/./before-
..... tl h570

```

**File Key:** a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

```

\c__hook_generic_include./after_-
 tl h570
\c__hook_generic_include./before_-
 tl h570
\c__hook_generic_include./end_-
 tl h570
\c__hook_generic_package./after_-
 tl h570
\c__hook_generic_package./before_-
 tl h570
\c__hook_generics_file_prop
 h546, h592
\c__hook_generics_prop
 h476, h504, h570, h587, h589
\c__hook_generics_reversed_iि-
 prop h484, h507, h592
\c__hook_generics_reversed_iिि-
 prop h487, h510, h592
__hook_gput_code:nmn h323, h324, h325
__hook_gput_next_code:nn
 h1087, h1088, h1088
__hook_gput_next_do:nn
 219, h384, h398, h1088, h1094, h1098
__hook_gput_next_do:Nnn
 h1088, h1100, h1103
__hook_gput undeclared_hook:nnn
 218, h366, h366, h377, h393
__hook_gremove_code:nn
 h601, h602, h603, h628
__hook_gset_rule:nnnn
 h636, h638, h641, h646
\c__hook_hash_tl 264,
 266, 267, i11, i176, i177, i179, i264
\g__hook_hook_curr_name_tl
 214–216, 249,
 250, h32, h204, h214, h255, h267,
 h282, h283, h290, h300, h301, h321
__hook_hook_gput_code_do:nnn ...
 h115, h323, h332, h342, h369
__hook_if_declared:n h1253
__hook_if_declared:nTF
 208, 209, h72, h154, h506, h1253, i66
__hook_if_declared_p:n h1253
__hook_if_deprecated_generic:n ..
 h1276
__hook_if_deprecated_generic:nTF
 h434, h625, h643, h950, h1268
__hook_if_deprecated_generic:w ..
 h1277, h1278
__hook_if_deprecated_generic_-
 p:n h1268
__hook_if_disabled:n h130
__hook_if_disabled:nTF
 208, 209, 211,
 h121, h151, h336, h961, h1034, h1090
__hook_if_disabled_p:n h121
__hook_if_execute_immediately:n
 h1212
__hook_if_execute_immediately:nTF
 h327, h1193, h1212
__hook_if_execute_immediately_-
 p:n h1212
__hook_if_file_hook:w
 h520, h522, h534, h536
__hook_if_file_hook:wTF
 219, 222, 223, 242, h402, h520, h1175
__hook_if_file_hook_p:w h520
__hook_if_generic:n h1268
__hook_if_generic:nTF
 h421, h525, h958, h1268
__hook_if_generic:w .. h1269, h1270
__hook_if_generic_p:n h1268
__hook_if_generic_reversed:n h1288
__hook_if_generic_reversed:nTF .
 h159, h429, h1288
__hook_if_generic_reversed:w ...
 h1289, h1290
__hook_if_generic_reversed_p:n .
 h1288
__hook_if_has_hash:n i232
__hook_if_has_hash:nTF
 264, 266, i166, i232
__hook_if_has_hash:w
 i232, i233, i234, i240, i242
__hook_if_has_hash_check:w
 i232, i239, i244
__hook_if_has_hash_p:n i232
__hook_if_label_case:nnnnn
 h702, h702, h779, h1048
__hook_if_public_command:N 261, i107
__hook_if_public_command:NTF i76, i86
__hook_if_public_command:w
 i76, i110, i116
__hook_if_reversed:n h1259
__hook_if_reversed:nTF
 h743, h977, h1014, h1016, h1259
__hook_if_reversed_p:n h1259
__hook_if_structure_exist:n h1247
__hook_if_structure_exist:nTF ..
 208, 209,
 h93, h605, h1093, h1227, h1247, h1387
__hook_if_structure_exist_p:n h1247
__hook_if_usable:n h1241
__hook_if_usable:nTF
 208, 209, 225,

```

**File Key:** a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

```

h330, h345, h423, h478, h621, h732,
h959, h976, h1032, h1214, h1241, h1454
__hook_if_usable_p:n . h1007, h1241
__hook_if_usable_use:n
.... 242, h1162, h1177, h1180, h1182
__hook_include_legacy_code_-
chunk:n h88, h109, h109, h731
__hook_init_structure:n
..... 208, 210, 218,
245, h85, h91, h91, h352, h368, h650
__hook_initialize_all:
..... h710, h710, h1525
__hook_initialize_hook_code:n ..
.. 233, h711, h727, h727, h1132, h1159
__hook_initialize_single:Nnn ...
... 228, 229, 231, h749, h763, h763, h833
\l__hook_label_0_tl h758
__hook_label_if_exist_apply:nnnTF
... 233, 234, 239, h836, h838, h840, h843
__hook_label_ordered:nn .. 228, h694
__hook_label_ordered:nnTF
..... 227, h663, h669, h675, h694
__hook_label_ordered_p:nn ... h694
__hook_label_pair:nn
..... 227, 228, h662, h668, h674, h678,
h681, h685, h686, h686, h918, h919
\l__hook_labels_int
... 233, h758, h766, h770, h802, h823
\l__hook_labels_seq
..... h758, h765, h771, h789, h924
__hook_list_if_rule_exists:nnnTF
..... h1041, h1058, h1059, h1061
__hook_list_one_rule:nnn
..... h1041, h1050, h1051, h1056
__hook_list_rules:nn
..... 239, h994, h1041, h1041, h1075
__hook_log:nN
..... h934, h937, h942, h948, h953
__hook_log_cmd:n
..... h936, h941, h945, h947, h957
__hook_log_line:n
..... h934, h944, h960, h962,
h966, h973, h985, h992, h1010, h1029
__hook_log_line_indent:n
..... h934, h946,
h968, h971, h979, h986, h997, h1005
__hook_log_next_code:n
..... h990, h1039, h1039
__hook_make_name:n 207, 214, h195,
h201, h210, h216, h216, h270, h312
__hook_make_name:w h216, h218, h222
__hook_make_prefixes:w
..... i153, i204, i224, i229
__hook_make_usable:n
.... 220, h76, h79, h79, h157, h427, h482
__hook_msg_pair_found:nnn
..... h861, h868,
h875, h883, h891, h902, h915, h915
\g__hook_name_stack_seq
..... h32, h256, h257,
h261, h268, h282, h289, h297, h307
__hook_new:n ... h68, h69, h70, h104
__hook_new_reversed:n
..... h100, h101, h102
__hook_next<hook> ... 207, 225, 244
__hook_normalize_hook_args:Nn ..
... h69, h101, h124, h148, h223, h232,
h937, h942, h1087, h1112, h1194, h1484
__hook_normalize_hook_args:Nnn .
..... h223, h237, h324, h602
__hook_normalize_hook_args_-
aux:Nn h223, h223, h234, h239, h247
__hook_normalize_hook_rule_-
args:Nnnnn h223, h245, h638
\l__hook_param_text_tl
..... 264, 265, i8, i172, i193, i221
__hook_parse_dot_label:n
..... h174, h176, h176
__hook_parse_dot_label:w
..... h176, h186, h189
__hook_parse_dot_label_aux:w ...
..... h176, h192, h200
__hook_parse_dot_label_cleanup:w
..... h176, h196, h199
__hook_parse_label_default:n ...
..... h170, h170,
h235, h241, h242, h249, h250, h252
__hook_patch_check:NNnn
..... i76, i80, i83, i86, i96
__hook_patch_cmd_or_delay:Nnn ..
... 220, 259, 260, i28, i41, i45, i45, i55
__hook_patch_command:Nnn
..... 260, i55, i73, i76, i76
__hook_patch_debug:n
..... i17, i17, i78, i79, i82,
i85, i88, i155, i286, i322, i325, i326, i331
__hook_patch_DeclareRobustCommand:Nnn
..... 262, i125, i129, i129
__hook_patch_DeclareRobustCommand_-
aux:Nnn i131, i134
__hook_patch_expand_redefine:NNnn
..... 262,
268, i140, i145, i150, i153, i153, i288
__hook_patch_newcommand:Nnn ...
..... 262, i126, i139, i143, i143
\l__hook_patch_num_args_int ...
..... i7, i156, i161, i164, i178

```

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxpld.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspac.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,
X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

```

\l_hook_patch_prefixes_tl 265, i8, i203, i219
\l_hook_patch_required_catcodes: 271, i336, i336, i357, i383
\l_hook_patch_retokenize:Nnnn 269, i327, i362, i362
\l_hook_preamble_hook:n 241, 242, h725, h956, h1115, h1121, h1131, h1140, h1158, h1186, h1198
\l_hook_rear_tl h758, h787, h793, h794, h816, h817
\l_hook_redefine_with_hooks:Nnnn 265, i153, i208, i217
\l_hook_replace_text_tl . 264, i8, i173, i180, i181, i182, i187, i194, i214
\l_hook_retokenize_patch:Nnn i91, i284, i284
\l_hook_return_tl h25, h289, h290, h297, h301, h358, h361, h617, h803, h804
\l_hook_rule_<_gset:nnn h660
\l_hook_rule_>_gset:nnn h660
\l_hook_rule_after_gset:nnn h660, h666, h671
\l_hook_rule_before_gset:nnn 232, h660, h660, h665
\l_hook_rule_gclear:nnn 227, h651, h683, h684
\l_hook_rule_incompatible-error-_gset:nnn h677
\l_hook_rule_incompatible-warning-_gset:nnn h677
\l_hook_rule_unrelated_gset:nnn 227, h683, h683
\l_hook_rule_voids_gset:nnn h672, h672
\l_hook_seq_csname:n h756, h757, h773, h807, h864, h871, h929
\l_hook_set_default_hook_label:n h305, h305, h1419
\l_hook_set_default_label:n h305, h312, h314
\l_hook_str_compare:nn h23, h23, h688, h696, h705
\l_hook_strip_double_slash:n h554, h560, h561
\l_hook_strip_double_slash:w h554, h562, h563, h567
\l_hook_tl_csname:n h756, h762, h772, h788, h791, h793, h797, h809, h811, h814, h816, h821, h862, h863, h869, h870, h928
\l_hook_tl_gclear:N h65, h65, h67, h117, h610, h611, h615, h798, h1207, h1208, h1209
\l_hook_tl_gput:Nn 233, h744, h746, h804, h830, h834, h834
\l_hook_tl_gput_left:Nn h58, h58, h63, h744
\l_hook_tl_gput_right:Nn h55, h55, h57, h353, h746, h831, h1109
\l_hook_tl_gset:Nn h46, h46, h48, h50, h52, h53, h54, h56, h60, h662, h668, h674, h678, h681, h736, h1108, h1204
\l_hook_tl_gset_eq>NN .. h64, h64, h66
\l_hook_tl_set:Nn 206, h40, h40, h42, h44, h45, h772
\l_hook_tmp:w 271, h34, h34, h258, h262, h264, h1043, h1064, h1073, h1084, i170, i176, i177, i179, i348, i351, i355, i365, i368, i381
\l_hook_tmpt_bool 238, h24, h993, h996, h1004, h1013
\l_hook_tmpt_t1 h25, h268, i163, i165, i167, i290, i305, i308, i354, i357, i380, i383
\l_hook_tmptb_t1 . h25, i295, i306, i309
\l_hook_toplevel_(hook) 207, 218, 225, 244
\l_hook_try_declarng_generic_-hook:nmm 218, h338, h371, h371, h373, h388, h390
\l_hook_try_declarng_generic_-hook:nNNnn 218, 219, h392, h397, h400, h400
\l_hook_try_declarng_generic_-hook:wn h416, h418, h468, h470, h496, h498
\l_hook_try_declarng_generic_-hook:wnTF 220, h375, h382, h411, h416, h463
\l_hook_try_declarng_generic_-hook_split:nNNnn h400, h404, h407, h409
\l_hook_try_declarng_generic_-next_hook:nn 219, h371, h380, h395, h1095
\l_hook_try_file_hook:n 242, h1162, h1170, h1173
\l_hook_try_patch_with_catcodes:Nnnnw 268, i302, i319, i319, i332
\l_hook_try_put_cmd_hook:n h426, h481, i20, i20, i22, i31, i33
\l_hook_try_put_cmd_hook:w i20, i23, i24, i34, i35

```

**File Key:** a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltruatex.dtx, e=lxpexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspc.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfnctcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\\_\\_hook\\_unpatchable\\_cases:n i407, i409  
 \\_\\_hook\\_update\\_hook\\_code:n . . . . .  
     . . . . . 217, 225, 228, h333, h622,  
     h655, h709, h709, h711, h715, h1106  
 \\_\\_hook\\_use:wn . . . . . 242, h1143,  
     h1156, h1162, h1162, h1165, h1167  
 \\_\\_hook\\_use\\_end: . . . . .  
     . . . . . h1115, h1151, h1153, h1160  
 \\_\\_hook\\_use\\_initialized:n . . . . .  
     241, h724, h1115, h1125, h1145, h1200  
 \\_\\_hook\\_use\\_once:n h1191, h1194, h1196  
 \\_\\_hook\\_use\\_once\\_clear:n . . . . .  
     . . . . . 243, h1201, h1203, h1205  
 \\_\\_hook\\_use\\_once\\_set:n . . . . .  
     . . . . . 243, h1199, h1203, h1203  
 \\_\\_hook\\_use\\_undefined:w . . . . .  
     . . . . . h1115, h1149, h1153  
 \g\\_hook\\_used\\_prop . . . . .  
     . . . . . h31, h712, h719, h752  
 \l\\_hook\\_work\\_prop . . . . .  
     . . . . . 231, h30, h748, h768, h775,  
     h777, h786, h803, h827, h896, h907  
 hook\\_\\_?? internal commands:  
     \\_\\_hook\\_\\_?? . . . . . 226  
 hook\\_\langle hook\rangle internal commands:  
     \\_\\_hook\\_\langle hook\rangle . . . . . 207, 211, 229  
 \hookleftarrow . . . . . B480  
 \hookrightarrow . . . . . B478  
 hook ?? internal commands:  
     \\_\\_hook~?? . . . . . h633  
 \phantom . . . . . I75  
 \hrule . . . . . b469, b513,  
     p350, p358, p384, p392, s289, s294,  
     B340, B618, K163, K168, K216,  
     K226, L359, L376, M590, M600, P395  
 \hrulefill . . . . . b513, f779, f800  
 \hsize . . . . . 1036  
 \hskip . . . . . 995, 1009, 1023, 1030, 1047  
 \hskip... . . . . . 1032  
 \hspace . . . . . p437, p441, p446  
 \hss . . . . . 890  
 \Hwithstroke . . . . . 370, 1047, s494, s1213  
 \hwithstroke . . . . . 370, 1047, s510, s1214  
 \hyphenation . . . . . 1016, 1019, 1021, s205  
 \hyphenchar . . . . . 105, 1045,  
     f737, f740, f749, f752, f755, f760, H454  
 \hyphenpenalty . . . . . b321, v661, v693

**I**

\I . . . . . b404, T1220,  
     T1352, T1442, T1462, Z241, Z541  
 \i . . . . . 1038,  
     s247, s402, s452, s453, s454, s455,  
     s456, s457, s547, s587, s588, s680,

s682, s684, s686, s778, s1140, s1295,  
     s1297, s1299, s1301, s1352, s1355,  
     s1358, s1361, s1431, Z245, Z545, Z556  
 \ialign . . . . . 997, b501, b503,  
     B337, B459, B530, B533, B537,  
     B540, L191, M141, I155, I157, I176  
 \IeC . . . . . Z343, Z347, Z454  
 \if . . . . . 619, 1031  
 if commands:  
     \if:w . . . . . 244, h1217, h1262  
     \if\_case:w . . . . . g1241, h688, h705  
     \if\_catcode:w . . . . . 267, i269  
     \if\_charcode:w . . . . . g838, h1292  
     \if\_cs\_exist:w . . . . .  
         . . . . . h448, h845, h1063, h1127, h1147  
     \if\_false: . . . . . g773, g776  
     \if\_int\_compare:w . . . . . h696, n59  
     \if\_meaning:w . . . . . g1155, i261, i264, V241  
     \if\_mode... . . . . . 303  
     \if\_mode\_horizontal: . . . . . n29  
 \if... . . . . . 841  
 \IfBlankF . . . . . g2625  
 \IfBlankT . . . . . g2625  
 \IfBlankTF . . . . . 1060, g2625, R19  
 \IfBold . . . . . 531  
 \IfBooleanF . . . . . g2600, t82, t90  
 \IfBooleanT . . . . . g2600  
 \IfBooleanTF . . . . . 178, g2600  
 \IfClassAtLeastTF . . . . . 794  
 \IfClassAtLeastTF . . . . . T165  
 \IfClassLoadedTF . . . . . 794  
 \IfClassLoadedTF . . . . . T261  
 \IfClassLoadedWithOptionsTF . . . . . 794  
 \IfClassLoadedWithOptionsTF . . . . . T261  
 \ifcsname . . . . .  
     . . . . . 445, 452, f635, f652, w440, w443,  
     w653, w656, x128, z153, A42, A53,  
     A75, A86, T1109, T1240, W422, X1903  
 \ifdefined . . . . . b537, b601, e22, e71, e72,  
     e73, e74, e117, e118, e119, A623, Z305  
 \iff . . . . . B500  
 \IfFileExists . . . . . 330, 795  
 \IfFileExists . . . . . 348, 1004, a181, e37,  
     e68, e114, r414, r426, r528, r584,  
     r609, T835, V149, V185, V195, Z572  
 \iffontchar . . . . .  
     . . . . . s866, s1060, s1165, s1167, s1169, s1216  
 \IfFontSeriesContextTF . . . . .  
     . . . . . 531, A496, A532, A534  
 \IfFormatAtLeastTF . . . . . 794  
 \IfFormatAtLeastTF . . . . . 625, 859, T165  
 \IfHookEmptyTF . . . . .  
     . . . . . 185, 186, 189, 251, h1453, h1556, H210

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

```

\IfHookExistsTF 251, h1454, h1555
\ifinclsname 845
\ifinner P57, P126, P315, I264, I272, I292, I319
\ifmmode 992
\IfNoValueF g2619
\IfNoValueT g2619
\IfNoValueTF g2619
\ifnum 875
\ifodd z1064, M320, M344, M378, M400, P68, P137, X21, X138, X610, X668, X982, X985, X1018, X1021, X1132, X1135, X1294, X1297, X1574, X1577, X1695, X1698, X1818, X2073, X2081
\IfPackageAtLeastTF 794
\IfPackageAtLeastTF T165
\IfPackageLoadedTF 794
\IfPackageLoadedTF T261
\IfPackageLoadedtTF T264, T272
\IfPackageLoadedWithOptionsTF 794
\IfPackageLoadedWithOptionsTF T261
\IfPDFManagementActiveTF 990, Z631
\IfTargetDateBefore T1670
\IfValueF g2622
\IfValueT g2622
\IfValueTF g2622
\ifvbox X319, X376, X423, X502, X518
\ifvoid 1033
\ifx 104, 266, 519, 521, 535, 861, 1045, 1050, 1054
\ignorespaces ... 201, 358, 1029, p49, p143, p162, p174, p185, p201, p214, p483, r71, r138, r193, s72, v292, v302, v312, H216, H223, H274, H289, H337, H342, H348, J55, J217, K154, K384, K401, K418, L57, L62, L68, L83, L92, L105, L109, L116, L123, L125, L134, L154, L237, L301, L303, L305, L332, M36, M46, M66, M75, M109, M120, M131, M143, M148, M154, N30, N32, O110, P17, P24, P477, P495, P513, R7, R9, W328, I300, I327
\ignorespacesafterend H7
\IJ 1040, 1058, s250, s436, s550, s1141, Z569
\ij 1040, 1058, s249, s434, s549, s1142, Z569
\Im B314
\imath B309
\immediate 202, 995
\in B429, B461
in commands:
 in_callback d798
 \in_callback 48
 \include 330
 \include 199, 201, 340, 848, 850, 851, 855, r217, r265, r267, r283, r285, r340
 include/.../after 851
 include/.../before 851
 include/.../end 851
 include/after 851
 include/before 851
 include/end 851
\IncludeInRelease 30, 75, 108, 1042, 1053, 1056, 1060, a21, a26, a293, a310, b49, b88, b103, b119, b125, b134, b139, b148, b154, b168, b182, b186, b220, b228, b233, b244, b289, b347, b357, b421, b439, b524, b548, b582, b589, b615, b655, b660, b670, c68, c177, c181, c185, c189, d3, d235, d258, d281, e2, e11, e20, e58, e112, e133, e145, e156, e166, e175, e189, f5, f13, f56, f62, f272, f314, f342, f370, f377, f394, f410, f448, f460, f488, f499, f511, f518, f614, f631, f661, f702, f721, f732, f749, f760, f771, f791, g983, g1159, g1162, g1306, g2625, g2631, g2659, h121, h139, h145, h164, h371, h388, h416, h468, h496, h520, h534, h554, h557, h570, h587, h592, h595, h1115, h1134, h1162, h1165, h1399, h1404, h1529, i20, i31, i386, i389, i419, l138, l158, l192, l205, n125, o19, p5, p21, p54, p65, p81, p86, p98, p104, p132, p152, p166, p178, p191, p206, p235, p252, p271, p307, p333, p366, p426, p432, p440, p445, p459, p465, r10, r83, r141, r219, r256, r277, r292, r348, r409, r419, r445, r465, r483, r493, r509, r534, r539, r547, r561, r572, r588, r621, s77, s104, s148, s169, s324, s332, s353, s369, t24, t30, t46, t69, t77, t99, t126, t161, t177, t185, t203, u5, u11, v24, v53, v210, v232, v284, v295, v305, v420, v431, v439, v471, v479, v493, v514, v550, v637, v699, w2, w385, w392, w404, w416, w424, w506, w522, w600, w613, w623, w630, w638, w710, w729, w735, w739, x113, x158, x161, x541, x550, y2, y22, z49, z78, z138, z217, z247, z278, z357, z365, z698, z746, z792, z801, z985, z996, A33, A68, A101, A126,

```

**File Key:** a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=lptra.dtx, o=ltmeta.dtx, p=ltspacel.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

A228, A252, A301, A348, A377,  
 A395, A435, A469, A498, A531,  
 A543, A577, A590, A621, A630,  
 A655, A672, A688, A705, B63, B81,  
 B100, B110, B622, B634, D27, D34,  
 E607, G38, G51, G62, H10, H65,  
 H107, H112, H124, H151, H170,  
 H226, H276, H301, H309, H322,  
 H328, H340, H345, H355, H374,  
 H391, H415, H437, H464, H489,  
 H502, H517, H533, H556, H566,  
 J125, J133, K4, K14, K50, K70,  
 K91, K102, K117, K125, K181,  
 K189, K235, K244, K281, K303,  
 K371, K389, K405, K425, K431,  
 K451, K459, L60, L65, L137, L157,  
 L222, L227, M10, M16, M26, M39,  
 M55, M69, M80, M89, M100, M112,  
 M146, M151, M160, M172, M235,  
 M250, M311, M369, M455, M464,  
 M473, M501, M531, M558, M572,  
 M585, M596, M608, M625, M645,  
 M652, M686, M746, M804, M820,  
 O5, O20, O161, O168, O174, O186,  
 O193, O202, O225, P35, P105,  
 P206, P232, P280, P294, P335,  
 P352, P406, P412, P420, P427,  
 P434, P440, P446, P463, P482,  
 P499, P549, P559, R14, R25, R61,  
 R77, S20, S40, T18, T23, T36,  
 T51, T63, T87, T95, T101, T126,  
 T144, T167, T174, T185, T198,  
 T224, T243, T263, T271, T282,  
 T298, T312, T330, T343, T359,  
 T379, T402, T418, T447, T459,  
 T491, T509, T529, T547, T566,  
 T576, T586, T598, T630, T641,  
 T655, T665, T721, T751, T778,  
 T810, T956, T1036, T1043, T1097,  
 T1228, T1359, V5, V15, V27, V89,  
 V144, V181, V192, V205, V249,  
 V257, V269, V300, V323, V344,  
 V358, V366, V389, V409, V434,  
 V480, V492, V503, V530, W3,  
 W419, W459, W473, X24, X54,  
 X151, X179, X345, X366, X371,  
 X419, X591, X651, X794, X812,  
 X873, X894, X930, X954, X1066,  
 X1217, X1386, X1468, X1562,  
 X1684, X1898, X1921, X1935,  
 X1963, X2194, X2212, X2231,  
 X2277, Z8, Z16, Z23, Z38, Z57, Z66,  
 Z73, Z92, Z129, Z152, Z185, Z274,  
 Z279, Z300, Z325, Z334, Z427, I79,  
 I87, I109, I118, I139, I144, I152,  
 I163, I203, I220, I259, I267, I277,  
 I304, I386, I395, I432, I444, I456, I465  
 \includeonly ..... 330  
 \includeonly ..... 338, 851, r217, r257, r259, r278, r279  
 \indent ... 297, 305, 1032, p462, J161, L81  
 \IndentBox ..... 295, 297, n49, n129  
 \index ..... 779  
 \index ..... 1034, 1048,  
 O182, Q6, Q18, S24, S32, X620, X679  
 \indexentry ..... Q15  
 \inf ..... 125  
 \infty ..... B316  
 \initcatcodetable ..... d91  
 \input ..... 330, 795  
 \input ..... 187, 201, 792,  
 848, 849, 853, 855, 861, 994, 1027,  
 1040, a71, a177, a180, a237, d18,  
 e108, e130, f22, r544, x16, y106,  
 A751, A761, A771, B10, B11, B12,  
 B13, B14, B23, B41, B42, B46, B47,  
 B136, B137, B138, B139, B654,  
 B655, B656, E1129, T615, Z150,  
 Z164, Z189, Z267, Z311, Z391, Z577  
 \input@path ..... 1, 6  
 input@path commands:  
     \input@path: ..... a242  
 \inputencodingname 1046, Z368, Z390, Z472  
 \InputIfFileExists ..... 330, 795  
 \InputIfFileExists 347, 348, 818, 820,  
 855, 857, 868, 1004, 1055, o6, r527,  
 r550, r565, r575, r585, r639, s1546,  
 v391, A743, A753, A763, E845,  
 E1212, T921, T987, V143, Y8, Z261  
 \inputlineno ..... 1006, a330, l214  
 \insert .....  
     293, b254, b279, b281, b284, b299,  
     P465, P484, P502, X517, X518, X1883  
 \int ..... B348  
 int commands:  
     \int\_add:Nn ..... g520, g1104  
     \int\_compare:nNnTF .... g388, g515,  
     g770, g812, g1692, g1995, g1997,  
     h791, h813, h823, h1305, i161, i292,  
     z194, W55, W111, W141, W351, W357  
     \int\_decr:N .....  
     ..... g467, g481, g495, g1691, h802  
     \int\_eval:n .....  
     e160, g2009, h810, h863, h870, W353  
     \int\_gdecr:N ..... V556  
     \int\_gincr:N .. g967, V546, W87, W102  
     \int\_gset:Nn ..... z199, V542

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=lt hyphen.dtx, Z=ltfinal.dtx

```

\int_incr:N . 140, g292, g411, g784,
 g849, g856, g1071, g1302, g1701, h770
\int_new:N g15, g27, g35,
 g1671, h759, i7, V541, W344, W346
\int_set:Nn g989, g999, g1168,
 g1177, g1227, g1675, g2102, g2128,
 i156, i199, i342, i343, V212, V226
\int_step_inline:nnn i164, i178
\int_use:N 875,
 g25, g815, g971, g999, g1089, g1093,
 g1094, g1177, g1293, z182, z183,
 z185, z194, V550, W104, W116, W361
\int_value:w W48, W136
\int_zero:N g283,
 g377, g690, g693, g819, g1063, h766
\c_max_int . W212, W238, W263, W290
\c_zero_int i161, W91
\interdisplaylinepenalty
 p13, I55, I194, I380
\interfootlinepenalty b394
\interfootnotelinepenalty
 b394, p18, P467, P486, P504
\interlinepenalty 994, p11, v663, H426,
 H429, H447, H450, O67, O118,
 O209, O232, P467, P486, P504,
 X338, X1153, X1157, X1319, X1323
\inteval 76
\inteval 76, e155, e170
\intextsep X1136, X1140,
 X1155, X1158, X1165, X1298,
 X1304, X1321, X1324, X1334, X2337
\intop B347, B348
\iotaota B276
iow commands:
 \iow_char:N
 ... g399, g463, g548, g617, g629,
 g660, g1009, g1015, g1676, g2411,
 g2501, h893, h904, h1320, h1350,
 h1362, h1367, h1372, h1519, h1521,
 i397, n111, n118, n119, n120, n121
 \iow_log:n h936
 \iow_newline: g1193, g1203, g1300, V554
 \iow_now:Nn W360
 \iow_term:n h344, h718, h720,
 h729, h806, h825, h826, h828, h892,
 h903, h917, h923, h924, h925, h928,
 h932, h941, h1072, h1077, i18, i27,
 i38, i47, i49, i65, i69, V547, V580
 \ishortstack M132
\itdefault w697, A30, B94
\item 292, 652, 994, 996,
 1000, 1001, 1012, l280, H351, H401,
 H403, H417, H439, J141, J219, L78,
 N36, N38, R4, R8, I439, I451, I478
\itemindent
 ... 651, 652, J9, J42, J95, J187, J208
\itemitem 994
itemize (env.) J242
\itemize J242
\itemsep 652, J1, J176
\iterate a84, a85, b457
\itshape 453,
 533, 994, 1024, 1039, s447, s807,
 w695, w696, A28, A29, A554, A587,
 A593, D21, E634, N36, N38, P399

```

**J**

```

\J Z243, Z543
\j s248, s403,
 s548, s779, s1150, s1365, s1451, Z556
\jmath B310
\jobname 826, 858
\Join A725
\joinrel B471, B478, B480, B482, B484,
 B486, B488, B490, B492, B496, B498
\jot I53, I191, I391, I401

```

**K**

```

\k 376, 1012, 1022,
 1039, s485, s590, s595, s617, s622,
 s698, s699, s757, s758, s812, s814,
 s819, s821, s1251, s1319, s1320,
 s1337, s1338, s1360, s1361, s1362,
 s1415, s1416, s1449, s1450, E181, E204
\kanjiskip e74
\kappa B277
\ker I27
\kern 887, 1009
\kern... 1032
kernel internal commands:
 __kernel_chk_if_free_cs:N
 ... 166, 180, g2065, g2066, g2656
 __kernel_cmd_if_xparse:NTF
 ... 137, 138, 143, 261, 1056,
 g1019, g1224, g2058, g2160, g2188, i127
 __kernel_exp_not:w
 ... h41, h47, h49, h56, h61
 \l__kernel_expl_bool i297
 __kernel_file_name_sanitize:n . V98
 __kernel_msg.... 1056
 __kernel_msg_.... 1058
\kerneltmpDoNotUse 269, 270,
 i335, i347, i353, i358, i364, i371, i384
keys commands:
 \keys_define:nn U35,
 U44, U116, U124, U128, U148, U179
 \keys_if_exist:nnTF
 ... U33, U75, U83, U95

```

**File Key:** a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\l\_keys\_key\_str ..... U40, U133  
 \keys\_set:nn ..... U47, U194  
 \l\_keys\_usage\_load\_prop ..... U144  
 \l\_keys\_usage\_preamble\_prop .. U175  
 keys internal commands:  
   \\_\_keys\_option\_end: .. U29, U43, U50  
   \\_\_keys\_options:Nn .....  
     ..... U24, U24, U137, U139  
   \\_\_keys\_options:nN ... U24, U25, U26  
   \\_\_keys\_options\_class:n U64, U71, U71  
 \l\_\_keys\_options\_clist .... 843,  
   U22, U30, U47, U77, U85, U97, U108  
   \\_\_keys\_options\_end: ..... U24  
   \\_\_keys\_options\_expand\_module:Nn  
     . U25, U117, U117, U124, U128, U194  
   \\_\_keys\_options\_expand\_module:nN  
     ..... U117, U121  
   \\_\_keys\_options\_global:Nn .....  
     ..... U31, U59, U59  
   \\_\_keys\_options\_loaded:n .....  
     ..... U51, U142, U142  
   \\_\_keys\_options\_loaded:nn .....  
     ..... U142, U151, U156  
 \l\_\_keys\_options\_loading\_bool ...  
   ..... 843, U23, U46, U48, U158  
   \\_\_keys\_options\_local: .....  
     ..... U32, U102, U102  
   \\_\_keys\_options\_package:n .....  
     ..... U67, U91, U91  
 \c\_\_keys\_props\_root\_str .... U9, U10  
   \\_\_keys\_remove\_equals:n .....  
     ..... U84, U95, U113, U113  
   \\_\_keys\_remove\_equals:w .....  
     ..... U113, U114, U115  
   \\_\_keys\_tmp:nn ..... U5, U11, U13  
 \l\_\_keys\_tmpa\_tl ..... U144, U146  
 \kill ..... L154, L162

**L**

\L ..... s242, s424, s530, s771, s1143,  
   T1217, T1349, T1439, T1461, Z569  
 \l ..... s251, s426, s551, s780, s1144, Z569  
 \label ..... 778,  
   1034, G32, O182, S24, S32, X619, X678  
 \labelenumi ..... 663  
 \labelenumiv ..... 663  
 \labelformat ..... 610  
 \labelformat .. G46, G52, G58, G63, G69  
 \labelitemi ..... 663  
 \labelitemii ..... 663  
 \labelitemiii ..... 663  
 \labelitemiv ..... 663  
 \labelsep 652, 663, J9, J210, J216, N36, N38

\labelwidth .....  
   .. 650, 652, J9, J93, J209, J211, J214  
 \Lambda ..... B300  
 \lambda ..... B278  
 \land ..... B368, B370  
 \langle ..... B594  
 \language ..... 1045, b35, b82, b84,  
   b99, r52, r121, H422, H562, X597, Y10  
 \lastbox ..... 299, v681, J130, J136,  
   J185, O99, O132, X305, I180, I181  
 \LastDeclaredEncoding .....  
   ..... 1036, 1039, v137, v140, Z468  
 \lastnamedcs ..... f653  
 \lastnodetype ..... v674, v675, v676, v680  
 \lastpenalty ..... v677, D112, D115  
 \lastskip 995, b481, b482, b484, b486,  
   p44, p128, p140, p159, p220, p221,  
   p225, p227, p228, p240, p258, p280,  
   p283, p315, p318, p319, D102,  
   D105, J115, J116, J150, J151, M123  
 \LaTeX ..... f732, q3, q15,  
   T1179, T1312, T1401, W386, W392  
 \LaTeXe ..... q13  
 \latexrelease ..... 868  
 \latexreleaseversion ..... c1  
 \lbrace ..... s308, B598  
 \lbrack ..... b408  
 \lccode ..... 1039, 1042, l19, l20,  
   l21, l22, l23, l24, s140, s1065, H507,  
   H522, H537, H580, Z210, Z227,  
   Z235, Z242, Z244, Z245, Z247,  
   Z249, Z250, Z251, Z252, Z527,  
   Z535, Z542, Z544, Z545, Z547, Z549  
 \lceil ..... B602  
 \lceil ..... 1033  
 \ldotp ..... B501, B504, B619  
 \ldots ..... 1021, s322, B505  
 \le ..... B416, B418  
 \leaders ..... b513, B340,  
   B558, B559, B561, B562, L376,  
   M565, M578, M590, M600, O214, O237  
 \leadsto ..... A728  
 \leavevmode ..... 996, 1008, 1016, 1022,  
   1023, 1029, 1032, 1034, 1035, 1039,  
   b472, b499, b502, b513, b515, p407,  
   p421, s75, s184, s289, s290, s393,  
   s425, s429, s432, s479, s763, s796,  
   D123, E98, E927, H426, H447,  
   H460, H471, H479, H558, H568,  
   H581, J58, J103, K8, K17, K24,  
   K156, K158, K174, K202, K263,  
   K339, K438, K455, K462, L178,  
   M134, M314, M373, O40, O210,

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

O222, O233, P530, R34, X157,  
 X162, X184, X189, I439, I451, I478  
 \left .. B625, B627, B629, B631, B636,  
 B637, B638, B639, I154, I160, I182  
 \Leftarrow .. B410, B492, B498  
 \leftarrow .. B437, B439, B480, B490, B496, B550  
 \leftarrowfill .. B534, B550  
 \lefteqn .. I415  
 \leftharpoondown .. B453, B467  
 \leftharpoonup .. B452  
 \lefthyphenmin .. Y11  
 \leftline .. K489  
 \leftmargin .. 650, 652, J9, J52, J53, J94, J146, J148  
 \leftmargini .. 652, J17, I431  
 \leftmarginii .. J17  
 \leftmarginiii .. J17  
 \leftmarginiv .. J17  
 \leftmarginv .. J17  
 \leftmarginvi .. 652, J17  
 \leftmark .. 1001, S48  
 \Leftrightarrow .. B409  
 \leftrightarrow .. B436  
 \leftskip .. b494,  
 v658, H359, H365, H369, H379,  
 H383, H387, H419, H441, J74,  
 K295, K316, O207, O212, O230, O235  
 \legacyoldstylenums .. E4, E617  
 \leq .. B414, B416  
 \let .. 30, 31, 91, 94,  
 251, 408, 556, 560, 868, 894, 994,  
 995, 1002, 1011–1013, 1023, 1026, 1028  
 \LetLtxMacro .. 98  
 \lfloor .. B606  
 \lg .. I4  
 \lgroupt .. B608  
 \lhd .. A731  
 \lhook .. B477, B478  
 \lim .. I6  
 \liminf .. I8  
 \limits .. B539, B543, I149, I340  
 \limsup .. I7  
 \line .. I269, M158, M450, M809, M826  
 \linebreak .. 310  
 \linebreak .. p9, p26  
 \linepenalty .. b320  
 \lineskip .. b402, b467, b502, B458, K297,  
 K317, L71, L198, M136, M315,  
 M374, W221, W272, X622, X681, I187  
 \lineskiplimit ..  
 ... 1045, b403, b467, b504, b505,  
 B458, B510, K283, K298, K305,  
 W222, W273, X622, X681, I189, I193  
 \linespread .. v316  
 \linethickness .. M130, M810, M827  
 \linewidth .. 651, r30,  
 r99, r157, J15, J51, J52, J54, K293,  
 K314, L36, P266, X146, X205,  
 I285, I311, I440, I452, I479, I483, I502  
 list (env.) .. J34  
 \list .. J34, J236, J247  
 \listfiles .. 795  
 \listfiles .. 202, 1011, 1016, r654  
 \listparindent .. 652, J9, J41, J50  
 \literal .. 1001, 1015  
 \ll .. B434  
 \llap .. J238, J249, K493  
 \lmoustache .. B563  
 \ln .. 15  
 \lnot .. B326, B327  
 \LoadClass .. 793  
 \LoadClass .. 997, T621, T651,  
 T869, T1004, T1078, T1086, T1087  
 \LoadClassWithOptions .. 793  
 \LoadClassWithOptions .. T650  
 \LoadFontDefinitionFile .. v418,  
 v444, v445, B21, B27, B28, B29, B33  
 \LoadPackageWithOptions .. 849  
 \loccount .. d17  
 \log .. I3  
 \loggingall .. b524  
 \loggingoutput ..  
 ... 1038, b520, b538, b556, b572, b586  
 \LogHook .. 190, h1442  
 \long .. 90, 1015,  
 1016, 1022, 1024, 1025, 1037, 1052  
 \Longleftarrow .. B492  
 \Longleftarrow .. B489  
 \Longleftarrow .. B498, B500  
 \Longleftarrow .. B496  
 \longmapsto .. B494  
 \Longrightarrow .. B486  
 \Longrightarrow .. B487, B494  
 \loop .. 1019, a84, b457, d150,  
 d159, v672, L382, T1127, T1188,  
 T1258, T1321, T1372, T1410, Z351,  
 Z362, Z372, Z383, Z413, Z439, Z449  
 \looseness .. b333  
 \lor .. B369, B371  
 \lower .. q2, B458,  
 K214, M35, M45, M196, M305,  
 M306, M353, M354, M409, M410  
 \lowercase .. 1031, l26, s141, s1066, s1544,  
 v332, v390, H511, H526, H541, H581  
 \lq .. b406  
 \rbox (env.) .. 666  
 \rbox .. K144

**File Key:** a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\ltfilehookdate ..... V539 \mathalpha ... z883, z1062, B169, B170,  
 \ltfilehookversion ..... V539 B171, B172, B173, B174, B175,  
 lua commands:  
   \lua\_now:n ..... W31 B176, B177, B178, B179, B180,  
 \luabytecode ..... d202 B181, B182, B183, B184, B185,  
 \luachunk ..... d210 B186, B187, B188, B189, B190,  
 \luadef ..... 44, d182, d186 B191, B192, B193, B194, B195,  
 \luafunction ..... 44, d178, d182, d186 B196, B197, B198, B199, B200,  
 luatexbase ..... d288 B201, B202, B203, B204, B205,  
 luatexluafunction ..... a21, a26 B206, B207, B208, B209, B210,  
 \luatexversion ..... a14, d5, e73, e119, s994 B211, B212, B213, B214, B215,  
                        B216, B217, B218, B219, B220,  
                        B221, B222, B223, B224, B225,  
                        B226, B227, B228, B229, B230,  
                        B297, B298, B299, B300, B301,  
                        B302, B303, B304, B305, B306,  
                        B307, B516, B517, B518, B519,  
                        B520, B521, B522, B523, B525, B528  
 M  
 \M ..... b404 \mathbf ..... A14,  
 \Macro ..... 994 A255, A308, A353, A381, A475, B151  
 \magstep ..... b395 \mathbin ..... z1067, B232, B233, B235, B358,  
 \magstephalf ..... b395 B359, B360, B361, B364, B365,  
 \makeat... ..... 1013 B366, B367, B370, B371, B372,  
 \makeatletter ..... 1009, 1034, 1035, B373, B374, B375, B376, B377,  
                        f728, i293, r32, r101, r159, v396, B378, B379, B380, B381, B382,  
                        H26, H79, O151, T615, T833, T965, X2 B383, B384, B385, B386, B387,  
 \makeatother ..... 1009, f728, i293, T615, Z633 B388, B389, B390, B391, B392,  
 \makebox ..... 666 B393, B394, B395, B396, B397, I37  
 \makebox ..... K3, I285, I311 \mathcal ..... B150  
 \makeglossary ..... 779 \mathchar ..... 1044,  
 \makeglossary ..... 1018, r204, Q20 b500, z824, z868, B335, B336, B617  
 \makeindex ..... 779 \mathchardef .....  
 \makeindex ..... 1018, r203, Q3    ... 1017, b21, b22, b23, b24, b107,  
 \makelabel ..... 651, 994, b110, b111, d227, j3, j4, j5, j6, s70, z859  
                        995, J45, J97, J205, J218, J238, J249  
 \MakeLowercase ..... 1025, Z562, Z571 \mathcharzero ..... d227  
 \MakeRobust .....  
   ... 510, 1052, 1056, f271, f773, f774, \mathchoice ..... I61  
   f775, f776, f777, f778, f779, f780, \mathclose ..... z1070, B231,  
   f781, f782, f783, f784, f785, f786, B240, B242, B245, B250, B256,  
   f787, f788, z796, z991, M806, M807, B258, B260, B566, B593, B597,  
   M808, M809, M810, M811, M812, B601, B605, B611, I43, I46, I49, I52  
   M813, M814, M815, M816, M817 \mathcode ..... z856, B252, B253, B254  
 \maketitle ..... 743 \mathdollar ..... 1017, s307, B614  
 \MakeUppercase ..... 1025, 1030, G48, G60, Z551 \mathellipsis ..... 1017, s321, B619  
 \mapsto ..... B444 \mathfontset ..... 992  
 \mapstochar ..... B443, B444, B494 \mathgroup ..... 992,  
 \marginpar ..... 295, P308 b79, v14, x303, x309, x315, x316,  
 \marginparpush ..... X85, X1834 x327, B643, E8, E14, E614, E1139  
 \marginparsep ..... X84, X1845, X1847 \mathhexbox ..... 1024, b500, A652  
 \marginparwidth ..... P341, P359, X83, X1847 \mathindent ..... 1003,  
 \mark ..... 30, 1001, S27, S35, S53 1015, 1050, I429, I441, I453, I481, I492  
 \markboth ..... S21, S22, S41, S43 \mathinner ..... B504, B508, B513, B619  
 \markright ..... S22, S44 \mathit ..... w696, A29, B153, B156, B617  
 \marks ..... d37, Z10, Z12 \mathnormal ..... B149  
 \math (env.) ..... I332 \mathop ..... z1066, B341, B342,  
 \math ..... I332 B343, B344, B345, B346, B347,  
 \mathaccemt ..... 1044  
 \mathaccent ..... z713, z761, z795, z805

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

B349, B350, B351, B352, B353,  
 B354, B356, B357, B537, B540, I3,  
 I4, I5, I6, I7, I8, I9, I10, I11, I12,  
 I13, I14, I15, I16, I17, I18, I19, I20,  
 I21, I22, I23, I24, I25, I26, I27, I28,  
 I29, I30, I31, I32, I33, I34, I149, I340  
 $\mathopen{}$  ..... z1069, B241, B244, B249, B255,  
 B257, B259, B564, B595, B599,  
 B603, B607, B609, I41, I44, I47, I50  
 $\mathord$  ..... z883, z1065, B236,  
 B243, B246, B251, B263, B264,  
 B265, B267, B268, B269, B270,  
 B271, B272, B273, B274, B275,  
 B276, B277, B278, B279, B280,  
 B281, B282, B283, B284, B285,  
 B286, B287, B288, B289, B290,  
 B291, B292, B293, B294, B295,  
 B296, B308, B309, B310, B311,  
 B312, B313, B314, B315, B316,  
 B317, B318, B319, B320, B321,  
 B322, B323, B324, B325, B327,  
 B328, B329, B330, B331, B332,  
 B333, B334, B524, B526, B527,  
 B549, B550, B553, B554, B555,  
 B556, B568, B570, B572, B575,  
 B577, B591, B613, B614, B615, B616  
 $\mathpalette$  ..... B457, B461, B464, I60, I69, I99, I129  
 $\mathparagraph$  ..... 1017, 1033, s310, t168, t180, B614  
 $\mathpunct$  ..... z1071, B234, B238, B501, B502, B503  
 $\mathrel$  ..... z1068, B237, B239,  
 B247, B248, B261, B262, B338,  
 B398, B399, B400, B401, B402,  
 B403, B404, B405, B406, B407,  
 B408, B409, B410, B411, B414,  
 B415, B418, B419, B420, B421,  
 B422, B423, B424, B425, B426,  
 B427, B428, B429, B430, B432,  
 B433, B434, B435, B436, B437,  
 B438, B441, B442, B443, B445,  
 B446, B447, B448, B449, B450,  
 B451, B452, B453, B454, B455,  
 B457, B461, B464, B471, B473,  
 B476, B477, B479, B482, B484,  
 B579, B581, B583, B585, B587,  
 B589, I42, I45, I48, I51, I149, I340  
 $\mathring$  ..... 1035, B528  
 $\mathrm$  . 993, A5, A406, A449, A481, B148  
 $\mathsection$  ..... 1017, 1033, s311, t167, t179, B614  
 $\mathsf$  A8, A411, A454, A484, B152, B155  
 $\mathsterling$  ..... 1017, s319, B614  
 $\mathstrut$  ..... 184, I93, I158, I159  
 $\mathsurround$  ..... 1034, b488  
 $\mathsymbol$  ..... z861  
 $\mathtt$  ..... A11, A416, A459, A487, B154  
 $\mathunderscore$  .. 1018, 1020, 1034, B614  
 $\mathversion$  . 993, 1010, v336, A614, A616  
 $\mathversion.$  ..... 992  
 $\matrix$  ..... I156, I160, I167  
 $\max$  ..... I22  
 $\maxdeadcycles$  ..... X7  
 $\maxdepth$  ..... 1005, b376, p287, r60, r128, r183,  
 X92, X169, X170, X506, X514,  
 X546, X715, X724, X764, X991, Z138  
 $\maxdimen$  . 893, 994, 1013, b309, b377,  
 b378, b467, b505, b521, b536, b537,  
 b555, b571, b586, v645, v655, v690,  
 v705, x384, x437, B458, M475,  
 M503, M533, M611, M628, T1482,  
 T1523, T1532, W349, W351, W398,  
 X291, X1853, X1873, X1878,  
 X2199, X2239, X2240, X2242, Z142  
 $\mbox$  ..... 666  
 $\mbox$  . 994, 1022, b500, q13, s293, s409,  
 s568, s1165, A648, B506, K11, K20,  
 K24, M52, P409, P416, P437, P444  
 $\mddefault$  ..... 516, 517,  
 520, 522, 1050, 1058, A18, A284,  
 A290, A291, A292, A331, A332,  
 A333, A342, A369, A385, A402,  
 A443, A479, B92, B105, B107, B121  
 $\mdseries$  204, 523, 531, A16, A17, A221,  
 A277, A328, A329, A363, A364,  
 A383, A384, A477, A478, A651, D20  
 $\mdseries$  ..... A420  
 $\mdseries/defaults$  ..... A420  
 $\meaning$  ..... a222, a231, a326, f228, f290, f328,  
 f356, f439, f699, z535, z548, z649,  
 z714, z761, z825, z919, z1015, z1119  
 $\medbreak$  ..... b482, f780, f801  
 $\mediumseries$  ..... 523  
 $\medmuskip$  B645, I36, I38, I211, I214, I228  
 $\medskip$  ..... b485, p400  
 $\medskipamount$  ..... b484, p401, p403  
 $\medspace$  ..... I201  
 $\MessageBreak$  ..... 1014,  
 1019, 1034, c98, e78, e81, e82,  
 e83, e84, e85, e86, e99, e100, e101,  
 e102, e103, f204, f279, f318, f346,  
 l3, l6, l13, l33, l46, l60, l73, l220,  
 l222, l228, l235, o12, s161, s988,  
 s1549, s1552, v34, v35, v536, v570,

**File Key:** a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=lt hyphen.dtx, Z=ltfinal.dtx

w456, x20, x21, x67, x88, x327, x478, x498, x530, x546, x561, x574, y31, y33, z181, z490, z499, z637, A61, A94, D144, E23, E79, E81, E100, E851, E853, E854, E855, E857, E859, E860, E861, E862, E863, E913, E915, E922, E929, E1144, H43, H83, T290, T304, T677, T688, T690, T692, T703, T864, T865, T866, T867, T877, T878, T880, T881, T882, T884, T886, T972, T973, T975, T976, T977, T979, T981, T999, T1000, T1001, T1002, T1063, T1080, T1081, T1149, T1166, T1205, T1280, T1299, T1338, T1393, T1427, T1536, T1538, T1620, T1623, T1636, T1638, W99, W173, W466, W467, W468, W469, X578, X1994, X2031, Z286, Z287, Z288, Z290  
\mho ..... A724  
\mid ..... B402  
\min ..... I23  
\minipage (env.) ..... 667  
\minipage ..... K326  
\mit ..... A776  
\mkern ..... B335, B338, B340, B462, B471, B513, B514, B515, B545, B546, B547, B548, B549, B550, B551, B552, O215, O238, I36, I37, I40, I73, I74  
mlist commands:  
\list\_to\_hlist ..... d847  
mode commands:  
\mode\_if\_horizontal:TF ..... n53, n58  
\mode\_if\_inner:TF ..... n54  
\mode\_if\_math:TF ..... z211  
\mode\_if\_vertical:TF ..... n72, n84  
\models ..... 1038, B484  
module commands:  
\module\_error ..... d345  
\module\_info ..... d345  
\module\_warning ..... d345  
\module\_error ..... 47  
\module\_info ..... 47  
\module\_warning ..... 47  
modules ..... d298  
\month a188, c11, c17, T1183, T1316, T1405  
\overright ..... X625, X684  
\mp ..... B389  
\mscount ..... L379  
msg commands:  
\msg\_... ..... 1056, 1058  
\msg\_error:nn ..... h274, h291, n80, n90  
\msg\_error:nnn ..... g277, g659, g2048, g2052, g2544, g2547, g2555, h73, h299, h337, h355, h1091, U182, V75  
\msg\_error:nnnn ..... g390, g398, g422, g426, g462, g476, g489, g572, g579, g588, g616, g628, g647, g681, g1006, g1579, g2116, g2120, g2511, g2525, g2567, g2581, i42, i100, i314, n18, n30, n66, U39, U163  
\msg\_error:nnnnn ..... g603, g1723, g1730, g2008, h278, h309, h318  
\msg\_error:nnnnnn ..... h657, h876  
\msg\_expandable\_error:nn ..... h180  
\msg\_expandable\_error:nnn ..... e182, g348, g2613, h208  
\msg\_expandable\_error:nnnn ..... g1900, g1927, V465  
\msg\_info:nnnn ..... g65, g76, g181, g185  
\msg\_line\_context: ..... g2468, g2473, g2478, g2483, h1325, h1330, h1363  
\g\_msg\_module\_name\_prop ..... h1300  
\g\_msg\_module\_type\_prop ..... g50, h1298, h1299  
\msg\_new:nnn ..... e184, g1007, g2410, g2465, g2470, g2475, g2480, g2485, g2497, h1323, h1328, h1359, h1365, h1370, h1375, h1380, h1384, h1391, h1517, U165, V471  
\msg\_new:nnnn ..... g2253, g2260, g2267, g2274, g2281, g2288, g2295, g2302, g2309, g2316, g2324, g2331, g2338, g2345, g2353, g2361, g2370, g2377, g2384, g2391, g2397, g2403, g2412, g2418, g2425, g2431, g2438, g2444, g2455, h1301, h1311, h1316, h1333, h1348, i391, i401, n102, n113, U53, U167, U187  
\msg\_redirect\_module:nnn ..... g49  
\msg\_warning:nnn ..... g73, h152  
\msg\_warning:nnnn ..... g640, h618, h630, h1516, U160  
\msg\_warning:nnnnn ..... h449  
\msg\_warning:nnnnnn ..... h884  
\mskip ..... I36, I38, I206, I225, I228, I230, I231  
\mu ..... B279  
\ubyte ..... Z338  
\multicolumn ..... 623, 1032, L233  
\multiput ..... M78, M811, M828  
\multispan ..... 1027, L233, L379  
\muskip ..... b29, b55, b93, d34, B545, B546  
\muskipdef ..... b55, b93, d228  
\muskipzero ..... d228

**File Key:** a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

**N**

\n ..... d332, d334, d341, d343, d470, d578, d603, d627, d672, d694, d713, d721, d722, d742, d755, d762, d763, d770, d782, Z107, Z112  
 \nabla ..... B319  
 \NAME ..... 79  
 \narrower ..... b493  
 \natural ..... B329  
 \ncallback ..... d677  
 \ndefault ..... d682, d686  
 \ne ..... 556, B413  
 \nearrow ..... B405  
 \NeedsFormat ..... 997  
 \NeedsTeXFormat ..... 1001, x12, E819, T682, T1682  
 \neg ..... B325, B326  
 \negmedspace ..... 1053, I201  
 \negthickspace ..... 1053, I201  
 \negthinspace ..... 1050, p468, I201  
 \neq ..... 556, B412  
 new commands:  
     new\_attribute ..... d411  
     new\_bytecode ..... d445  
     new\_chunkname ..... d458  
     new\_luafunction ..... d474  
     new\_whatsit ..... d433  
   \new\_attribute ..... 45  
   \new\_bytecode ..... 45  
   \new\_chunkname ..... 45  
   \new\_luafunction ..... 45  
   \new\_whatsit ..... 45  
   \newattribute ..... 44  
   \newattribute ..... d74, d238  
   \newbox ..... b47, b314, b490, j13, H482, J27, K115, L16, L17, L18, L343, M6, M670, M675, X86, X120, X121, X122, I66  
   \newcatcodetable ..... 44  
   \newcatcodetable ..... d84, d93, d94, d120, d121, d242  
   \newcommand ..... 79  
   \newcommand ..... 91, 94, 97, 100, f77, s4, w530, w535, w540, A36, A72, B51, B52, B53, B54, B56, B57, B59, B60, B92, B93, B94, B95, B96, B97, B120, B121, B122, H303, H304, H305, H306, M682, V570, V571, V572, V573, V575, X2325, X2328, X2331, X2332, X2335, X2336  
   \NewCommandCopy ..... 77, 94, 96, 97, 1052, 1057, 1058, f461, f463, g984, g1160  
   \newcount ..... 1040, b47, b352, b394, j7, j8, p124, r7, t36, x25, z27, z142, z377, J23, J24, J25, J26, J56, J226, J241, K367, L11, L12, L13, L14, L15, L335, L336, L337, M664, M665, M666, M667, M676, O36, O140, O141, P3, P267, P268, P269, P270, T1479, X103, X105, X107, X109, X111, X119, X2020, X2323, X2326, X2329, X2333, Z3, Z4, Z5, Z77, I55, I344, I345  
   \newcounter ..... 397  
   \newcounter ..... t10  
   \newdimen ..... b47, b309, b311, b312, b393, j10, j11, j12, p123, x398, x399, J9, J10, J11, J12, J13, J14, J15, J16, J17, J18, J19, J20, J21, J22, K171, K172, L3, L5, L6, L7, L8, L166, L338, L339, L340, L341, M3, M4, M5, M7, M431, M432, M433, M434, M435, M436, M668, M669, M671, M672, M673, M674, P451, X71, X72, X73, X75, X76, X77, X78, X79, X80, X81, X82, X83, X84, X85, X91, X93, X94, X106, X108, X110, X112, X113, X114, X115, X116, X117, X118, X2021, X2022, I53  
   \NewDocumentCommand ..... 76, 77, 94, 108, 167, 180, 257, 1050, g2505, h1396, h1397, h1398, h1401, h1402, h1410, h1412, h1414, h1416, h1418, h1430, h1432, h1446, h1448, h1451, t79, t87, U123, U125, U136, U138, U193  
   \NewDocumentEnvironment ..... g2284, g2291, g2541  
   \newenvironment ..... 80  
   \newenvironment ..... 199, f146, T1181, T1314, T1403  
   \NewExpandableDocumentCommand ..... g2561  
   \newfam ..... 993, 1024, b47, d38, v16  
   \newfont ..... A618  
   \newgroup ..... z47  
   \newhelp ..... b306  
   \NewHook ..... 181, 182, 189, 195, 197, 199, 200, 208, 209, 256, 257, h1396, h1532, r72, r73, r74, r343, x150, A420, A421, A422, A423, A424, A425, A426, A427, A428, H33, H34, H35, H36, H37, T950, T951, V177  
   \NewHookPair ..... 255  
   \newif ..... 992, c70, f168, f767, j9, r5, r6, v204, w401, w413, z15, A528, D82, E871, G3, J28, J29, J30, J31, J32, J33, J138, K423, L19, L251, M157, M427, M428, M429, M430, M459, M460, O38, O124,

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdfnns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspacel.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

T2, X95, X96, X97, X98, X99, X100, X101, X102, I75, I76, I190, I346  
 \newinsert ..... 1043, b193, b242, K368, P390, X27, X1852  
 \newlabel ..... 1026, G22, G34  
 \newlanguage ..... b47, Z270  
 \newlength ..... 405  
 \newlength ..... u3  
 \newline ..... p92, p99, p105  
 \newlinechar ..... a75, f20  
 \newluabytocode ..... 45  
 \newluabytocode ..... d197, d252  
 \newluachunkname ..... 45  
 \newluachunkname ..... d205, d254  
 \newluacmd ..... 44  
 \newluacmd ..... 45, d181  
 \newluafunction ..... 44  
 \newluafunction ..... 44, d4, d173, d236, d248  
 \newmarks ..... Z6  
 \newmathalphabet ..... y13, y109  
 \NewMirroredHookPair ..... 182, h1396, h1534  
 \NewModuleRelease ..... 1055, 1056, c149, g9, h4, i4, n4, o3, E2  
 \newmuskip ..... b47  
 \newpage ..... X133, X139, X150  
 \newprotectedluacmd ..... 45  
 \newprotectedluacmd ..... d181, s1010, W28  
 \newread ..... 995, b47, b307  
 \NewReversedHook ..... 182, 187, 195, 256, h1396, h1533, r344, r345, T952, T953, V178  
 \newrobustcmd ..... 94  
 \newsavebox ..... 666  
 \newsavebox ..... K115  
 \newskip b47, b310, b313, b391, b392, j14, j15, j17, p403, p404, p405, p454, u3, H400, J2, J3, J4, J5, J6, J7, J8, X2337, X2338, X2339, X2343, X2344, X2347, X2348, X2349, X2353, X2354, X2355, I347, I430  
 \newtheorem ..... 1027, N1  
 \newtie ..... 582, s851, E188, E189, E208, E705, E1015, E1016  
 \newtoks ..... 1043, b47, b306, j16, n35, v346, v347, w499, x247  
 \newwhatsit ..... 45  
 \newwhatsit ..... d189, d250  
 \newwrite ..... 1017, b47, b308, r3, r4, O154, Q4, Q21  
 \newXeTeXintercharclass ..... Z21  
 \next ..... 1009, 1020  
 nfss internal commands:  
   \\_\\_nfss\\_init\\_mv\\_freeze:N ..... z186, z210, z210, z211

**File Key:** a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

\normalfont .	203, 358, 523, 1023–1025, 1027, 1049, 1050, v646, v706, A653, A673, A675, A684, A689, A691, A699, A706, A708, A714, D18, H461, O218, O241, P401, I339, I426	
\normalfont . . . . .	A420	
\normalize . . . . .	1005	
\normallineskip .	b391, b402, K297, K317	
\normallineskiplimit . . . . .	b391, b403, K282, K298, K304, I193	
\normalmarginpar . . . . .	P387	
\normalsfcodes . . . . .	1034, r44, r46, r48, r113, r115, r117, r171, r173, r175, r197, X618, X677	
\normalshape . . . . .	448, 453, w682, w723	
\normalsize . . . . .	184, 1003, 1006, 1007, r42, r111, r169, D142, P23, P176, P372, T5, X617, X676	
\not . . . . .	B338, B412, B413, B435	
\notexpanded . . . . .	266	
\notin . . . . .	B461	
\noexpand . . . . .	997	
\nu . . . . .	B280	
\null . . . . .	1021, 1022, b416, s327, s363, s486, s489, s834, s837, s1253, G17, H426, H447, H558, H568, O212, O235, W374, I112, I121, I156, I185	
\nulldelimiterspace . . . . .	b380, B642	
\nullfont . . . . .	H167	
\number a89, c49, c58, d105, f2, f114, t142, v596, v599, x439, z64, z93, z113, z128, z164, z232, z262, z294, A645, T1094, T1183, T1316, T1405, W349		
\numberline . . . . .	996, O72, O82, O248, P17	
\numexpr . . . . .	b189, b205, b215, b246, d82, d105, d157, f642, s1034, z150, X36	
\nunknown . . . . .	d699	
\nwarrior . . . . .	B407	
\nxt . . . . .	1020	
<b>O</b>		
\o . . . . .	s244, s400, s533, s770, s1130, Z568	
\oalign . . . . .	b502	
\obeycr . . . . .	p480	
\obeyedline . . . . .	29, 30, b426, b430, b453	
\obeyedspace . . . . .	30, b431, b434, b454	
\obeylines . . . . .	29, 30, b419, f783, f804, H432, H453, H549, H550, X583	
\obeyspaces . . . . .	30, b419, f784, f805, X583	
\oddsidemargin . . . . .	X72, X74, X611, X670	
\odot . . . . .	B384	
\OE . . . . .	s243, s399, s532, s769, s1147, Z568	
\oe . . . . .	s252, s404, s553, s782, s1148, Z568	
<b>P</b>		
\P . . . . .	1012, 1017, s310	
package/.../after . . . . .	850	
package/.../before . . . . .	850	
package/after . . . . .	850	
package/before . . . . .	850	
\PackageError . . . . .	c74, c132, c144, l84, s1547, E825, E877, E921	
\PackageInfo . . . . .	l84, E829, E846, E851, E867, E868, E928, E1213	
\PackageNote . . . . .	1057, l136	
\PackageNoteNoLine . . . . .	l136	

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\PackageWarning . . . . .  
     . . . . . 184, E827, E878, E1142, W465  
 \PackageWarningNoLine . . 184, s986, X1993  
 \pagebreak . . . . . 310  
 \pagebreak . . . . . p6, p7, p22, p24  
 \pagegoal . . . . . X1880, X1887  
 \pagenumbering . . . . . 608  
 \pagenumbering . . . . . F5  
 \pageref . . . . . G10  
 \pageshrink . . . . . X538, X542, X558  
 \pagestyle . . . . . S2  
 \pagetotal . . . . . X128  
 \paperheight . . . . . X93  
 \paperwidth . . . . . X93  
 \par . . . . . 29–31,  
     293, 294, 296, 297, 302–306, 618,  
     660, 994, 996, 1030, 1051, 1055,  
     a123, b11, b412, b428, b430, b447,  
     b469, b478, b479, b480, b482, b484,  
     b486, d156, f9, f21, m3, m4, m5,  
     n98, n134, v671, H109, H165, H325,  
     H332, H424, H445, J63, J110, J127,  
     J129, J135, J161, J164, K288,  
     K309, K355, K385, K402, L195,  
     L385, O41, O90, O220, O242, P15,  
     P24, P249, P344, P478, P496, S62,  
     S63, W390, X166, X193, X257, X1886  
 para commands:  
     \para\_end: . . . . . 296,  
         303, 305, n51, n51, n98, n99, n100  
     \g\_para\_indent\_box . . . . .  
         297, 300, n14, n42, n44, n47, n49, n75  
     \para\_omit\_indent: . . . . . 297, 302, n46, n50  
     \para\_raw\_end: . . . . . 297, 305, n71, n93, n96  
     \para\_raw\_indent: . . . . .  
         . . . . . 297, 305, n71, n71, n94  
     \para\_raw\_noindent: . . . . .  
         . . . . . 297, 305, n71, n83, n95  
 para internal commands:  
     \\_\_para\_handle\_indent: . . . . .  
         . . . . . n31, n43, n43, n77  
     \g\_\_para\_standard\_everypar\_t1 . . . . .  
         . . . . . 301, 302, n12, n34, n36, n76, n87  
 para/after . . . . . 295, n6  
 para/before . . . . . 295, n6  
 para/begin . . . . . 295, n6  
 para/end . . . . . 295, n6  
 \paracntvalue . . . . . 298  
 \paragraphmark . . . . . O143  
 \parallel . . . . . B401  
 \parbox . . . . . 666  
 \parbox . . . . . 293, 295, K234  
 \parfillskip . . . . . 304, b390, v645, v660, v705,  
     H361, H371, H380, H388, H420,  
     H442, J76, K296, K317, O207, O230  
 \parindent . . . . .  
     292, b382, b494, b495, H361, H366,  
     H371, H380, H384, H388, H420,  
     H442, J50, K291, K312, O208, O231  
 \parsep . . . . . 652, J1, J49, J90  
 \parseunicodedataI . . . . . d123, d162  
 \parseunicodedataII . . . . . d124, d126  
 \parseunicodedataIII . . . . . d128, d134  
 \parseunicodedataIV . . . . . d130, d142  
 \parseunicodedataV . . . . . d146, d149  
 \parshape . . . . . 300, J54  
 \parskip . . . . . 292, 295,  
     299, 301, 652, 1019, b383, H336,  
     H418, H420, H440, H442, J49, J73,  
     J88, J90, J117, J153, J172, J223,  
     K291, K312, L79, X1159, X1327, I498  
 \partial . . . . . B315  
 \partopsep . . . . . 652, J1, J61, I496  
 \PassOptionsToClass . . . . . 793  
 \PassOptionsToClass . . . . . T378  
 \PassOptionsToPackage . . . . . 793  
 \PassOptionsToPackage . . . . . T378  
 \patterns . . . . . 1016, 1019, 1021, s205  
 \pausing . . . . . b334  
 \pdffilesize . . . . . e71, e117  
 \pdfgentounicode . . . . . Z301,  
     Z302, Z306, Z321, Z326, Z327, Z328  
 \pdfhorigin . . . . . W312  
 \pdftexrevision . . . . . Z307  
 \pdftexversion . . . . . Z305, Z306, Z307  
 \pdfvariable . . . . . W311, W316  
 \pdfvorigin . . . . . W317  
 peek commands:  
     \peek\_meaning:NTF . . . . . 170, g2212  
     \peek\_meaning\_remove:NTF . . . . .  
         . . . . . 155, 170, g1619, g1721, g2200  
     \peek\_N\_type:TF . . . . . g1625, g1649, g1681  
     \peek\_remove\_spaces:n . . . . . g1617  
 \penalty . . . . . 994, 1003, b474, b475,  
     b476, b477, b478, b479, b483, b485,  
     b487, p34, p37, p46, p281, p291,  
     p316, p320, D118, H426, H429,  
     H447, H450, J190, L56, P195,  
     P199, P201, P217, P221, P223,  
     R37, X136, X176, X195, X198,  
     X1157, X1323, I37, I194, I391, I401  
 \perp . . . . . B447  
 \phantom . . . . . I75  
 \Phi . . . . . B305  
 \phi . . . . . B287  
 \Pi . . . . . B302

**File Key:** a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\pi ..... B282  
 picture (env.) ..... M21  
 \picture ..... M21  
 \pm ..... B390  
 \pmatrix ..... I160, I168  
 \pmod ..... I39  
 \PopDefaultHookLabel ... 186, 187, h1418  
 \poptabs ..... l256, L142, L161  
 \poptracing ..... x148, x340  
 \postdisplaypenalty . p12, I438, I450, I476  
 \pounds ..... 1012, 1013, s318  
 \Pr ..... I32  
 pre commands:  
     pre\_shipout\_filter ..... 875  
 \prec ..... B421  
 \preceq ..... B424  
 \predisplaypenalty . b329, I437, I449, I475  
 \pretolerance .... b316, v647, v662, v707  
 \prevdepth ..... 322,  
     b465, b469, b470, p287, p288, p349,  
     p354, p383, p388, P196, P198,  
     P218, P220, X167, X169, X172, I192  
 \PreviousTotalPages .... 875, 893, W396  
 prg commands:  
     \prg\_break:n .....  
         .. g2603, g2605, g2607, g2609, g2610  
     \prg\_break\_point: ..... g2611  
     \prg\_do\_nothing: .....  
         ..... 133, g339, g851, g1562,  
         i190, i304, i422, V132, V134, W164,  
         W165, W170, W179, W183, W319  
     \prg\_new\_conditional:Npn .....  
         ..... h130, h522, h536, h694,  
         h1212, h1225, h1241, h1247, h1253,  
         h1259, h1268, h1276, h1288, i232  
     \prg\_new\_protected\_conditional:Npn .....

..... h418, h470, h498, i106, i345  
 \prg\_replicate:nn ..... g774,  
     g817, g862, g917, g2015, g2022, V549  
 \prg\_return\_false: .....  
     .... 220, h136, h440, h474, h492,  
     h502, h515, h529, h531, h544, h548,  
     h551, h699, h1220, h1223, h1237,  
     h1245, h1251, h1257, h1265, h1274,  
     h1283, h1286, h1295, i121, i245, i360  
 \prg\_return\_true: .....  
     ..... h135, h431, h490, h513,  
     h528, h547, h697, h1218, h1236,  
     h1239, h1244, h1250, h1256, h1263,  
     h1273, h1284, h1293, i120, i245, i359  
 \prime ..... B253, B317, I243  
 \ProcessedArgument .....  
     ..... 165, g320, g324, g331,

g1948, g1949, g1966, g1968, g1990,  
     g1999, g2005, g2013, g2030, g2638  
 \ProcessKeyOptions ..... 841, 842, U136  
 \ProcessKeyPackageOptions .....  
     ..... 841, 842, T828, U136  
 \ProcessList ..... g2643  
 \ProcessOption\* ..... 997  
 \ProcessOptions 842, 997, 1003, s1568,  
     x71, E844, E881, T472, T563, T1082  
 \ProcessOptions\* ..... T472  
 \prod ..... B349  
 prop commands:  
     \prop\_clear:N ..... g1470  
     \prop\_const\_from\_keyval:Nn .....  
         ..... h589, h597, h598, h599  
     \prop\_gclear:N ..... h609, h712, h1210, i57  
     \prop\_get:NnN ..... h803  
     \prop\_get:NnNTF ..... g1476, h358, U144  
     \prop\_gpop:NnNTF ..... h617  
     \prop\_gput:Nnn ..... g50, h360,  
         h363, h751, h1298, h1299, h1300, i50  
     \prop\_if\_empty:NTF ..... h734, h967  
     \prop\_if\_empty\_p:N ..... h1230  
     \prop\_if\_exist:NTF ..... h1249  
     \prop\_if\_in:NnTF ..... h476,  
         h484, h487, h504, h507, h510, h546  
     \prop\_map\_break: ..... h780, h1049  
     \prop\_map\_function:NN ..... i56  
     \prop\_map\_inline:Nn .....  
         ..... h719, h768, h775,  
         h777, h926, h970, h1044, h1046, U175  
 \prop\_new:N .....  
     ..... g45, h30, h31, h95, h633, h634, i16  
 \prop\_put:Nnn ..... g1467, h896, h907  
 \prop\_set\_eq:NN ..... h748  
 \prop\_show:N ..... 240  
 \proto ..... B398  
 \protect .....  
     ..... 624, 881, 883, 884, 994–996, 1008,  
     1013, 1014, 1018, 1019, 1023, 1055,  
     f102, f220, f234, f243, f248, f251,  
     f252, f254, f255, f260, f261, f266,  
     f269, f270, f295, f333, f361, f533,  
     f553, l246, l248, l250, l256, l262,  
     l269, l277, l280, l286, r210, s26, s32,  
     s51, s55, s209, s217, z598, z1146,  
     A638, D143, G12, H193, H203,  
     H238, H241, H256, H266, L264,  
     O12, O72, O82, O164, O171, O177,  
     P17, R5, V282, V305, W83, W95,  
     W124, W131, W151, X596, X655, Z344  
 \protected ..... 30, 94, 366, 453, 510,  
     1045, b430, d186, f7, f443, o5, p56,  
     p462, s308, s309, t194, w682, w686,

**File Key:** a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=lttoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=lt hyphen.dtx, Z=ltfinal.dtx

w689, w692, w695, w698, w701, w704, z988, A573, H126, H324, L56, L201, L208, M142, W485, W486, W487, Z552, Z553, I375, I413  
 \Provide ..... 1016  
 \providecommand ..... f178, h1541, h1544, s6, s981, E811, E812, X2004  
 \ProvideCommandCopy ..... 96  
 \ProvideDocumentCommand ..... g2505  
 \ProvideDocumentEnvironment ..... g2541  
 \ProvideExpandableDocumentCommand ..... g2561  
 \ProvideHook ..... 1056, h1487  
 \ProvideMirroredHookPair ..... h1487  
 \ProvideReversedHook ..... h1487  
 provides commands:  
     provides\_module ..... d299  
 \provides\_module ..... 47  
 \ProvidesClass ..... 793  
 \ProvidesClass ..... T358  
 \ProvidesExplPackage ..... V538  
 \ProvidesFile .....  
     ... 1010, 1019, 1024, 1025, 1027, a92, B665, B667, B668, B669, T367  
 \ProvidesPackage ..... 793  
 \ProvidesPackage ..... 804, 805, 1001, 1011, 1035, x13, E817, E849, T281, T360, T362, T1683, V567, W480  
 \ProvideTextCommand ..... 1016, s3, s60  
 \ProvideTextCommandDefault 1016, 1025, s57  
 \Psi ..... B306  
 \psi ..... B289  
 \PushDefaultHookLabel .. 186, 187, h1418  
 \pushtabs ... l256, L138, L139, L158, L160  
 \pushtracing ..... x117, x321  
 \put .... 872, 876, 878, 890, M56, M58, M70, M72, M325, M326, M327, M328, M336, M338, M351, M352, M353, M354, M361, M364, M384, M385, M386, M387, M393, M395, M407, M408, M409, M410, M415, M420, M729, M785, M813, M830

**Q**

\qbezier ..... 707  
 \qbezier ..... M682, M814, M831  
 \qbeziermax ..... M681, M707, M708, M768  
 \qqquad ..... p473  
 \quad ... 1023, p473, O111, I155, I157, I177  
 quark commands:  
     \q\_mark g1152, g1155, g1849, g1853, g1865, g2142, g2150, g2153, g2158  
     \q\_nil ..... 158-160, g514, g1144, g1155, g1253, g1376, g1397, g1401, g1407, g1416,

\quark\_if\_nil:NTF ... 160, g525, g1855  
 \quark\_if\_nil:nTF ..... g1250  
 \quark\_if\_recursion\_tail\_stop:N ..... g710, g1234  
 \quark\_if\_recursion\_tail\_stop:n ..... 410, g578, g1070, g2155, g2156, h260, U7  
 \quark\_if\_recursion\_tail\_stop\_- do:Nn ..... g533  
 \quark\_if\_recursion\_tail\_stop\_- do:nn ..... g436, g441, g450, g459, g473, g486, g500, g510, g554, i321  
 \quark\_new:N ..... g1385, i14, i15  
 \q\_recursion\_stop .....  
     ... g264, g275, g387, g700, g1066, g1230, g1483, g1487, g2141, g2144, g2150, g2151, h265, h266, i312, U19  
 \q\_stop g352, g514, g523, g528, g1152, g1154, g1401, g1418, g2006, g2171  
 quark internal commands:  
     \q\_cmd .....  
         ... 119, 158, 160, 161, g336, g340, g886, g1181, g1182, g1200, g1208, g1383, g1753, g1754, g1757, g1762, g1768, g1769, g1771, g1775, g1780, g1783, g1786, g1797, g1801, g1802, g1805, g1806, g1809, g1820, g1823, g1827, g1838, g1840, g1841, g1842, g1843, g1844, g1845, g1846, g1850, g1869, g1870, g1871, g1872, g1873, g1874, g1875, g1876, g1877, g1878, g1879, g1880, g1881, g1882, g1885, g1890, g1896, g1897, g1903, g1908, g1909, g1912, g1923, g1930, g1935, g1936, g1937, g1940, g1941, g1943  
     \q\_hook\_recursion\_stop ..... i14, i247, i248, i257, i278  
     \q\_hook\_recursion\_tail i14, i247, i261  
 \quotedblbase ..... s555, s783, s1176  
 \quotesinglbase ..... s556, s1173

**R**

\r ..... 1012, 1033, b410, b411, s236, s388, s431, s470, s610, s637, s647, s673, s756, s795, s1245, s1263, s1289, s1411, s1412, E183, E205  
 \radical ..... 1044, z1012, z1015, z1045  
 \raggedbottom ..... S53

**File Key:** a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\raggedleft . . . . . H367, H385, H396, H403  
 \raggedright . . . . . H362, H381, H395, H401  
 \raise . . . . . s327, s359, s430, s433, s732, s797, s895, s1253, A651, B465, B513, B515, K473, K482, M61, M73, M105, M117, M195, M305, M452, M495, M526, M551, M619, M636, M637, M740, M796, I73  
 \raisebox . . . . . 667  
 \raisebox . . . . . s872, s1222, K450  
 \rangleangle . . . . . B592  
 \RawIndent . . . . . 297, n94, n130  
 \RawNoIndent . . . . . n94  
 \RawNoindent . . . . . 297, n95, n131  
 \RawParEnd . . . . . 297, b449, n94, n132  
 \RawShipout . . . . . 871–874, 877, 879–882, W150, W427  
 \rbrace . . . . . s309, B596  
 \rbrack . . . . . b408  
 \rceil . . . . . B600  
 \Re . . . . . B313  
 \read . . . . . 348  
 \ReadonlyShipoutCounter . . . . .  
     875, 877, 897, W344, W429  
 \Ref . . . . . 610  
 \Ref . . . . . G47, G52, G59, G63, G70  
 \ref . . . . . 663, G10, G47, G59, P553  
 \refstepcounter . . . . . 397  
 \refstepcounter . . . . . 645,  
     1001, G39, G40, G52, G54, G63,  
     G65, J202, N27, O59, P9, I337, I477  
 \registernumber . . . . . 46  
 registernumber . . . . . d390  
 \relax . . . . . 89, 101, 293, 444, 623,  
     820, 861, 868, 894, 993, 994, 997,  
     998, 1010, 1016, 1019, 1021–1023,  
     1025–1028, 1031, 1037, 1046, 1056  
 \relax\_ . . . . . 1010  
 \Relbar . . . . . 1038, B476, B484, B486, B492  
 \relbar . . . . . B473, B488, B490  
 \relpenalty . . . . . b324  
 remove commands:  
     remove\_from\_callback . . . . . d753  
 \remove\_from\_callback . . . . . 47  
 \RemoveFromHook . . . . . 183, 184, 189, h1416, h1543  
 \removelastskip . . . . . b481, b483, b485, b487  
 \renewcommand . . . . . 79  
 \renewcommand . . . . .  
     96, 517, 524, 548, f124, B66,  
     B68, B70, B71, B73, B75, B77, B78,  
     B84, B86, B88, B89, B103, B104,  
     B105, B113, B114, I425, I445, I466  
 \RenewCommandCopy . . . . . 96, f461, f463  
 \RenewDocumentCommand . . . . . g2505  
 \RenewDocumentEnvironment . . . . . g2298, g2541  
 \renewenvironment . . . . . 80  
 \renewenvironment . . . . . f152, I474, I486  
 \RenewExpandableDocumentCommand . . . . .  
     177, g2561  
 \repeat . . . . . a84, a86, b457, d154,  
     d164, v688, L382, T1131, T1192,  
     T1262, T1325, T1376, T1414, Z355,  
     Z366, Z376, Z387, Z417, Z443, Z453  
 \requestedLaTeXdate . . . . .  
     T1477, T1510, T1530, T1616  
 \requestedpatchdate . . . . . T1540, T1617  
 \RequirePackage . . . . . 793  
 \RequirePackage . . . . .  
     187, 201, 849, 876, 997, d24,  
     T611, T618, T661, T670, T1078, X2001  
 \RequirePackageWithOptions . . . . . 793  
 \RequirePackageWithOptions . . . . . T653  
 reserved@**a** commands:  
     \reserved@a: . . . . .  
         r232, r304, r359, T1108, T1239  
 reserved@**b** commands:  
     \reserved@b: . . . . . T230, T247  
 reserved@**c** commands:  
     \reserved@c: . . . . . r658  
 \reservedb . . . . . 408  
 \RestoreAtCatcode . . . . . 1008, 1011  
 \restorecr . . . . . p480  
 \ReverseBoolean . . . . . g2639  
 \reversemarginpar . . . . . P387  
 \rfloor . . . . . B604  
 \rgroup . . . . . B608  
 \rhd . . . . . A733  
 \rho . . . . . B283  
 \rhook . . . . . B479, B480  
 \right . . . . . B625, B627, B629, B631, B636,  
     B637, B638, B639, I155, I160, I184  
 \Rightarrow . . . . . B411, B486, B498  
 \rightarrow . . . . . B438,  
     B440, B444, B478, B488, B496, B549  
 \rightarrowfill . . . . . B531, B547  
 \rightharpoondown . . . . . B455  
 \rightharpoonup . . . . . B454, B466  
 \righthyphenmin . . . . . Y11  
 \rightleftharpoons . . . . . B464  
 \rightline . . . . . K489  
 \rightmargin . . . . . 652, J9, J40, J51  
 \rightmark . . . . . 1001, S48  
 \rightskip . . . . . b495, v659, H359, H363,  
     H369, H379, H382, H387, H419,  
     H441, J75, K295, K316, O207, O230  
 \rlap . . . . . 1024,  
     s430, s433, s797, K493, L81, I415, I426

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

\rm ..... 1000  
 \rmdefault ..... 396, 522, 528, 529,  
     1037, A6, A208, A398, A407, A439,  
     A450, A482, B50, B120, E7, E613  
 \rmfamily ..... 203,  
     516, 521, 532, 1023, 1025, A4, A5,  
     A405, A448, A449, A480, A481, D15  
 rmfamily ..... A420  
 \rmmath ..... 993  
 \rmoustache ..... B565  
 \rmsubstdefault . B18, B30, E28, E39, E85  
 \Roman ..... 397  
 \Roman ..... 875, t138  
 \roman ..... 397  
 \roman ..... t137  
 \romannumeral ..... 624, t143, t144,  
     H192, H209, H255, J43, J234, J245  
 \root ..... I66, I343  
 \rootbox ..... I66  
 \rq ..... b406  
 \rule ..... 667  
 \rule ..... K384,  
     K401, K418, K424, P477, P495, P513

**S**

\S ..... 1017, s311  
 \samepage ..... 310  
 \samepage ..... 994, p11, p28  
 \SaveAtCatcode ..... 1008, 1011  
 \savebox ..... 666  
 \savebox ..... K116  
 \savecatcodetable .... d117, d168, d170  
 \sb ..... I199  
 \sbox ..... 666  
 \sbox ..... 1000,  
     1015, 1026, b489, q4, s498, s514,  
     J205, K122, K129, K133, K138, K143  
 scan commands:  
     \scan\_new:N ..... h38  
     \scan\_stop: .....  
         115, 302, 844, g1615, g1712,  
         g2171, h375, h382, h411, h419,  
         h451, h463, h471, h499, h1289,  
         h1290, i347, i348, i364, i365, n52,  
         U49, U61, W44, W323, W325, W326  
 scan internal commands:  
     \s\_file\_stop ..... V103, V105, V108, V110, V112, V125  
     \s\_hook\_mark .....  
         206, h38, h39, h186, h189,  
         h192, h196, h199, h200, h402, h444,  
         h446, h454, h456, h459, h461, h523,  
         h537, h562, h563, h567, h1143,  
         h1156, h1167, h1175, h1218, h1220,

h1263, h1265, h1269, h1270, h1277,  
     h1278, i23, i25, i34, i36, i112,  
     i117, i233, i244, i352, i355, i369, i381  
     \s\_keys\_stop ..... U114, U115  
 \scdefault ..... w694, A27, B94  
 \scriptfont ..... x338  
 \scriptscriptfont ..... x339  
 \scriptscriptstyle ..... I65, I68  
 \scriptspace ..... b381  
 \scriptstyle ..... B337, I64  
 \scshape .....  
     1034, s300, w692, w693, A25, A26, D23  
 \searrow ..... B406  
 \sec ..... I20  
 \secdef ..... O142  
 \secondoftwo ..... 344  
 \sectionmark ..... O143  
 \selectfont ..... 203, 204,  
     443, 445, 446, 452, 460, 461, 529,  
     538, 540, 993, 1010, 1054, 1056, q7,  
     s302, s329, s360, s449, s809, s871,  
     s1221, s1255, s1569, v291, v301,  
     v311, w528, w533, w538, w684,  
     w688, w691, w694, w697, w700,  
     w703, w706, x114, x115, x159,  
     x162, x164, A6, A9, A12, A15,  
     A18, A21, A24, A27, A30, A264,  
     A287, A325, A345, A360, A371,  
     A382, A385, A409, A414, A419,  
     A452, A457, A462, A476, A479,  
     A482, A485, A488, A564, A641,  
     A665, A682, A697, E36, E94, E104,  
     E636, E673, E916, E933, P403, P424  
 selectfont ..... x150  
 seq commands:  
     \seq\_clear:N ..... h765  
     \seq\_clear\_new:N ..... h773  
     \seq\_gpop:NNTF ..... h289, h297, V72  
     \seq\_gpop\_right:NN ..... h268  
     \seq\_gpush:Nn ..... h282, V64  
     \seq\_gput\_right:Nn .. h83, h257, h261  
     \seq\_if\_empty:NTF ..... h256, h307  
     \seq\_if\_exist:NTF ..... V60  
     \seq\_map\_inline:Nn .....  
         g1967, h713, h789, h807  
     \seq\_mapthread\_function:NNN ... 168  
     \seq\_new:N . g1951, h28, h33, h758, V61  
     \seq\_put\_right:Nn .. h771, h864, h871  
     \seq\_set\_split:Nnn ..... g1965  
     \seq\_use:Nnnn ..... h924, h929  
 \series ..... 993  
 \seriesdefault ..... 522, 523,  
     541, 1051, s1570, z349, A220, A222,  
     A660, A678, A694, A711, A773, B120

**File Key:** a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

```

\setattribute 45
\setattribute d82, d239
\setbox... 1032
\setbox0 1028
\setcounter 397
\setcounter ... 490, r390, t2, t37, z143,
 J225, X2324, X2327, X2330, X2334
\SetDefaultHookLabel 186, 187, 215, h1418
\SetKeys 842, U193
\setlength 405
\setlength 315, 324,
 1036, p83, p243, p442, u4, K43,
 K204, K265, K268, K341, K440,
 K441, K442, K471, K472, K479,
 K480, K481, L176, L384, X2340,
 X2341, X2342, X2345, X2346,
 X2350, X2351, X2352, X2356,
 X2357, X2358, I494, I499, I500, I501
\SetMathAlphabet
 .. v11, y140, y141, z603, B155, B156
\setminus B393
\setrangelccode .. d96, d104, d113, d114
\SetSymbolFont 997, z458, B145, B146, B147
\settodepth 405
\settodepth u17
\settoheight 405
\settoheight u17
\settowidth 405
\settowidth u17
\sf 1000
\sfcode b398, b399, b400, b401,
 b518, p416, r45, r114, r172, Z236, Z536
\sfdefault A9, A212,
 A399, A412, A440, A455, A485, B50
\sffamily . 203, 516, 529, 532, A7, A8,
 A410, A453, A454, A483, A484, D16
\sffamily A420
\sfsubstdefault B19, B31, E30, E87
\shape 993, 995
\shapedefault 448,
 523, 534, 549, 1049, s1570, w684,
 z350, A661, A679, A695, A712, B120
\sharp B330
\shipout 203, 871–874, 877,
 879, 881, 883, 884, 894, 897, 1014,
 W4, W52, W420, W423, X602, X660
shipout commands:
\l_shipout_box 872, 880, 882, 883,
 885, 887, W23, W35, W38, W50,
 W61, W125, W148, W177, W182,
 W205, W206, W213, W224, W227,
 W228, W232, W239, W249, W257,
 W264, W274, W280, W281, W284,
 W291, W293, W294, W300, W302
\l_shipout_box_dp_dim 872, W192,
 W195, W197, W228, W281, W501
\l_shipout_box_ht_dim 872, W191, W195,
 W197, W227, W247, W280, W500
\l_shipout_box_ht_plus_dp_dim ... 872, W194,
 W197, W213, W264, W275, W277
\l_shipout_box_wd_dim 872, W193, W197, W239, W291, W502
\shipout_debug_off: 876, W7, W13, W404
\shipout_debug_on: 876, W7, W8, W403
\shipout_discard: 874, W341, W341, W400
\g_shipout_READONLY_int 875, 881, W102, W104, W111,
 W116, W344, W353, W357, W361
\g_shipout_TOTALPAGE_int 875
\g_shipout_TOTALPAGES_int 875, 881, W87, W346
shipout internal commands:
__shipout_add_background_box:n .
 .. W178, W204, W204, W336, W409
__shipout_add_background_-
 picture:n W69, W335, W335, W413
__shipout_add_firstpage_-
 material:Nn
 .. W171, W187, W187, W402, W407
__shipout_add_firstpage_-
 specials: 882, 884,
 885, W110, W164, W176, W176, W179
__shipout_add_foreground_box:n .
 .. W117, W255, W255, W339, W411
__shipout_add_foreground_-
 picture:n W64, W338, W338, W415
__shipout_debug:n 878,
 W7, W7, W20, W103, W115, W147
\g__shipout_debug_bool
 .. W6, W10, W15, W21
__shipout_debug_gset:
 .. W7, W11, W16, W18
\g__shipout_discard_bool
 .. W81, W88, W90, W201, W342
__shipout_drop_firstpage_-
 specials: 884,
 885, W126, W165, W176, W181, W183
__shipout_excuse_extra_page: ...
 .. W373, W381, W381
__shipout_execute: 880, W46, W46, W52
__shipout_execute_cont:
 .. W57, W59, W59

```

**File Key:** a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

```

__shipout_execute_main_cont:Nnnn \shortstack M132, M147, M152, M815, M832
... 880, 883, W60, W77, W77, W146
__shipout_execute_nohooks_cont: \show 97, 99, 100, f505, f575, f576
... W143, W145
__shipout_execute_raw: show commands:
... 883, W134, W134, W150
__shipout_execute_test_level: .. \show_hook:n 236
... W49, W54, W54
__shipout_execute_test_level_- \showbox X1915
raw: W134, W137, W140
__shipout_finalize_box: \showboxbreadth b370,
... W26, W28, W44, W123
\l__shipout_firstpage_box b521, b598, b622, b639, b664, X1915
... 885, W168, W178, W185
__shipout_get_box_size:N \showboxdepth . b371, b521, b597, b621,
... 886, W85, W107, W190, W190, W205 b638, b665, v647, v691, v708, X1915
\l__shipout_group_level_tl \ShowCommand 94, 97, 99,
... W47, W53, W56, W135, W142
\c__shipout_horigin_tl . W308, W323 1052, 1057–1059, f498, g1163, g1307
__shipout_init_page_origins: ... \ShowDocumentCommandArgSpec g2644
... W308, W308, W319, W322
\g__shipout_lastpage_handled_- \ShowDocumentEnvironmentArgSpec .. g2644
bool W121, W186, W363
__shipout_picture_overlay:n ... \ShowFloat X1896
... W321, W321, W336, W339
\l__shipout_raw_box ... 882, W25, \ShowHook ... 190, 194, 195, h1442, h1548
... W138, W146, W148, W177, W182
__shipout_run_firstpage_hook: ... \showhyphens 1045, v636
... 882,
... 884, 885, W108, W161, W161, W170
\l__shipout_saved_badness_tl ... \showoutput 1025, b520
... W202, W208, W216,
... W229, W234, W242, W251, W259,
... W267, W279, W286, W296, W304
__shipout_saved_protect: ... \showoverfull
... W83, W131, W151, W151
\l__shipout_tmp_box 1025, b519, b522, b544, b579, b587
... W202, W215, W217, W218, W219,
... W223, W241, W243, W244, W245,
... W248, W266, W268, W269, W270,
... W276, W295, W297, W298, W299,
... W301, W327, W329, W330, W331
\c__shipout_vorigin_tl . W308, W325
shipout/after 872, W152
shipout/background 872, W152
shipout/before 872, W152
shipout/firstpage 872, W152
shipout/foreground 872, W152
shipout/lastpage 872, W152
\ShipoutBox 871–873, 877, 885, W23, W428, W483
\ShipoutBoxDepth W448, W500
\ShipoutBoxHeight 897, W447, W500
\ShipoutBoxWidth W449, W500
\shortstack M132, M147, M152, M815, M832
\show 97, 99, 100, f505, f575, f576
show commands:
\show_hook:n 236
\showbox X1915
\showboxbreadth b370,
b521, b598, b622, b639, b664, X1915
\showboxdepth . b371, b521, b597, b621,
b638, b665, v647, v691, v708, X1915
\ShowCommand 94, 97, 99,
1052, 1057–1059, f498, g1163, g1307
\ShowDocumentCommandArgSpec g2644
\ShowDocumentEnvironmentArgSpec .. g2644
\ShowFloat X1896
\ShowHook ... 190, 194, 195, h1442, h1548
\showhyphens 1045, v636
\showoutput 1025, b520
\showoverfull
... 1025, b519, b522, b544, b579, b587
\showtokens 101, f611
\Sigma B303
\sigma B284
\sim B445, B457
\simeq B446
\sin 19
\sinh I11
\size 993
\skew B544
\skip b28, b53, b92, b208, b250,
... b295, d33, K358, P391, X316, X490
skip commands:
\skip_eval:n e162
\skip_zero:N n23, W220,
... W221, W222, W271, W272, W273
\skipdef b45, b53, b92, d229
\skipeval 76
\skipeval 76, 1059, e155, e172
\skipzero d229
\slash b474, f785, f806
\sldefault w691, A24, B94
\SLiTeX 1015
\slippy K300, K319, S57, S62
\slloppypar (env.) S62
\slloppypar S62
\slshape s440,
... s800, w689, w690, A22, A23, D22, E627
\small 184
\smallbreak b482, f786, f807
\smallint B357
\smallskip b483, p400
\smallskipamount b482, p400, p403
\smash B473, B547, B548, B551, B552, I126
\smile B450

```

**File Key:** a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfnctcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\sourceLaTeXdate . . . . . c161, T65, T103  
 \sp . . . . . I199  
 \space . . . . . 30, 1009, 1011, 1014, 1022, b414  
 \spacefactor . . . . . 1036, b472, b473, p129,  
     p138, p157, p171, p183, p197, p211,  
     p416, p429, p434, s70, s73, P531, P533  
 \spacefactor\_in\_math\_mode\_gh/643 . . . . . 1059  
 \spaceskip . . . . . E6, E612  
 \spadesuit . . . . . B334  
 \span . . . . . L383  
 \special . . . . . 203, 873, 882, 884, 885, 897  
 \SplitArgument . . . . . g2639  
 \splitfirstmark . . . . . X2245  
 \SplitList . . . . . 180, g2639  
 \splitmaxdepth . . . . .  
     . . . . . b377, P469, P488, P506, X2239  
 \splittopskip . . . . . b389, P468, P487, P505  
 \sqcap . . . . . B376  
 \sqcup . . . . . B377  
 \sqrt . . . . . 1024, I342  
 \sqrtsign . . . . . 1024, B529, I71, I342  
 \sqsubset . . . . . A729  
 \sqsubseteq . . . . . B399  
 \sqsupset . . . . . A730  
 \sqsupseteq . . . . . B400  
 \SS . . . . . 1029, s304, s534, s1157, Z569  
 \ss . . . . . s254, s406, s557, s784, s1132, Z569  
 \sscdefault . . . . . w536, w609, w706  
 \sscsshape . . . . . w536, w608, w704, w705, D31  
 \stackrel . . . . . I340  
 \stamp . . . . . 1003  
 \star . . . . . B397  
 \stepcounter . . . . . 397  
 \stepcounter . . . . . 996, 1035, t17, t27,  
     v601, z48, G41, G54, G65, P452,  
     P520, X646, X705, I350, I410, I487  
 \stop . . . . . H165  
 \storedpar . . . . . d156, d161  
 str commands:  
     \c\_backslash\_str . . . . . i146, i315  
     \c\_hash\_str . . . . . g1293  
     \str\_case:Nn . . . . . g1010, i411  
     \str\_case:nNTF . . . . . g2490, i39  
     \str\_case\_e:nNTF . . . . . g2162  
     \str\_count:n . . . . . i158  
     \str\_gset:Nn . . . . . h1437  
     \str\_head:n . . . . . g2107  
     \str\_if\_eq:nn . . . . . 228  
     \str\_if\_eq:NNTF . . . . . V552  
     \str\_if\_eq:nnTF . . . . .  
         158, 161, e137, g357, g1050, g1055,  
         g1374, g1746, g1759, g1761, g1791,  
         g1811, g1813, g1832, g1859, g1887,  
         g1889, g1914, g1916, g1939, g2226,  
     h184, h276, h316, h348, h350, h425,  
     h480, h527, h539, h607, h614, h917,  
     h1000, h1080, h1169, h1304, h1336,  
     i209, i287, i373, i376, V131, V546, V556  
     \str\_if\_eq\_p:nn . . . . .  
         . . . . . g416, g417, g418, g419, h543, h800  
     \str\_map\_function:NN . . . . . e141  
     \str\_new:N . . . . . g19  
     \str\_set:Nn . . . . .  
         . . . . . g176, g177, g219, g2040, V400, V401  
     \str\_tail:n . . . . . 157, g1746, g2104, g2130  
     \str\_uppercase:n . . . . . g2426  
 str internal commands:  
     \\_\str\_if\_eq:nn . . . . . 204, h23  
 \strcmp . . . . . T283, T299  
 \stretch . . . . . p456  
 \string . . . . . 858, 861, 997, 999, 1014,  
     1028, 1031, 1036, 1043, 1049, 1055  
 \stripmeaning . . . . . 1000  
 \strut . . . . . 204, b490, f787, f808, L29, I178, I179  
 \strutbox . . . . . b490, x189,  
     K384, K401, K418, L186, L187,  
     P469, P477, P488, P495, P506, P513  
 \subparagraphmark . . . . . O143  
 \subsectionmark . . . . . O143  
 \subset . . . . . B426  
 \subseteq . . . . . B428  
 \substdefault . . . . . 1049  
 \subsubsectionmark . . . . . O143  
 \succ . . . . . B420  
 \succeq . . . . . B423  
 \sum . . . . . B350  
 \sup . . . . . I24  
 \supereject . . . . . 1027  
 \suppressfloats . . . . . X2006  
 \supset . . . . . B425  
 \supseteq . . . . . B427  
 \surd . . . . . B336  
 \swallow . . . . . B408  
 \swdefault . . . . . w531, w607, w703  
 \swshape . . . . . w531, w606, w701, w702, D30  
 \symbol . . . . . s162, A619  
 \symletters . . . . . E8, E14, E614, E1139  
 \symoperators . . . . . B643  
 sys commands:  
     \sys\_if\_engine\_luatex:TF . . . . . W26

## T

\T . . . . . l23, s335,  
     s337, s339, s341, s343, s345, s347,  
     s349, s351, s374, T1457, T1461, T1462  
 \t . . . . . 1017, s282, s741, s849, E157,  
     E158, E185, E189, E191, E203,  
     E206, E208, E688, E859, E1126, E1128

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

tabbing (env.) ..... L71  
 \tabbing ..... L71  
 \tabbingsep ..... L130, L132, L166  
 \tabcolsep ..... L259, L338  
 \tableofcontents ..... 618  
 \tabskip ..... 997,  
     b501, L167, L192, I195, I196, I356,  
     I359, I362, I364, I492, I505, I508, I510  
 tabular (env.) ..... L174  
 \tabular ..... L174  
 \tabular\* ..... L175  
 \tabularnewline ..... 1021, L194, L207  
 \tan ..... I15  
 \tanh ..... I17  
 \tau ..... B285  
 \tencirc ..... C10, M125, M678  
 \tencircw ..... C10, M128  
 \tenln ..... C9, M124, M126, M677, M679  
 \tenlnw ..... C9, M127, M129  
 \test ..... 1003  
 \TeX ..... 1048, q1, q12  
 TeX and L<sup>A</sup>T<sub>E</sub>X 2<sub>E</sub> commands:  
     \...-h@k ..... 819, 1048  
     \..\@without@substitution ..... 460  
     \@ ..... 998, 1041, a68, d20,  
         d862, f728, f729, i292, l19, p425,  
         q2, T43, T58, T71, T80, T118, Z503  
     \@...\hook ..... 210, 211  
     \@?\? ..... 998  
     \@o ..... a334, a335, k15, k19,  
         k20, k21, k22, k24, k27, k28, k30,  
         k31, r656, r672, x510, x512, x513,  
         L238, L239, L240, L250, X10, X11  
     \@{\@DeclareMathDelimiter} ..... 1009  
     \@{\@defaultsubs} ..... v585  
     \@{\@enc@update} ..... 1021, s183, v259, v263  
     \@{\@end} 348, 1003, a72, a225, f23, r633,  
         r634, H32, H99, H165, Y18, Z588, Z609  
     \@{\@endpbox} ..... L193, L236, L386  
     \@{\@eqncr} .. I368, I390, I400, I405, I511  
     \@{\@fileswith@pti@ns} T540, T559, T1014  
     \@{\@hyph} ..... f24, f745  
     \@{\@hyphenation} ..... s205  
     \@{\@if@newlist} X599, X644, X657, X703  
     \@{\@ifdefinable} ..... f132, s17  
     \@{\@input} ..... 1003, 1040,  
         a71, f22, r531, r544, r584, H26,  
         H79, T1604, V172, V188, V196, Z317  
     \@{\@italiccorr} .. 1005, f25, D113, D117  
     \@{\@line} ..... K489  
     \@{\@math@bgroup} ..... D131, D138  
     \@{\@math@egroup} ..... D128  
     \@{\@par} ..... 303, 306, 994, 1031, f21, m4, n98,  
         n135, H165, H421, H426, H429,  
         H443, H447, H450, J82, J85, K266,  
         K288, K309, L199, O67, O118, X257  
 \@{\@patterns} ..... s205  
 \@{\@protect} ..... f254, f260, f269  
 \@{\@sqrt} ..... 1027, 1030  
 \@{\@startpbox} ..... L193, L236, L386  
 \@{\@sverb} ..... 632, H500  
 \@{\@underline} ..... K445, K448, K449  
 \@{\@unprocessedoptions} .....  
     T926, T991, T1068  
 \@{\@warning} ..... 1001, l215  
 \@{\@Alph} ..... 104, t140, t156  
 \@{\@DeclareEncodingSubset} .....  
     E45, E47, E48, E49, E50, E53, E60  
 \@{\@DeclareMathDelimiter} .....  
     1009, z875, z894  
 \@{\@DeclareMathSizes} .. v206, v207, v209  
 \@{\@Ephack} . 995, p190, P201, P223, P241  
 \@{\@Ialph} ..... 1011  
 \@{\@IncludeInRelease} ..... c68  
 \@{\@IncludeInRelease} ..... c68  
 \@{\@M} .. b21, b475, b476, f37, f39, p11,  
     p12, p13, p14, p15, p16, p17, p18,  
     p119, v654, v661, x439, x452, J194,  
     L56, O67, O100, O118, O130, O209,  
     O232, X176, X195, X198, X258, I378  
 \@{\@MM} . 273, b21, P469, P488, P506, X299  
 \@{\@Mi} ..... j3, X136  
 \@{\@Mii} ..... j3, P53, P122, P194,  
     P216, P241, P311, X295, X1159, X1326  
 \@{\@Miii} .... j3, P55, P124, P313, X298  
 \@{\@Miv} j3, P195, P201, P217, P223, X272  
 \@{\@Roman} ..... 1034, t138, t144  
 \@{\@TeXversion} ..... 2, 1014, a329, l28  
 \@{\@abs@page@last} .....  
     891, 893, W105, W111, W349,  
     W351, W353, W361, W398, W399  
 \@{\@acci} ..... 1006,  
     1007, 1011, 1012, A775, K290, K311  
 \@{\@accii} ..... 1011, A775, K290, K311  
 \@{\@acciii} .... 1011, A775, K290, K311  
 \@{\@acol} L168, L178, L260, L261, L273,  
     L274, L277, L294, L309, L317, L327  
 \@{\@acolampacol} ..... L258,  
     L275, L277, L284, L292, L326, L329  
 \@{\@activechar@info} ..... 1008, X575  
 \@{\@activechar@warning} ..... 1008  
 \@{\@addamp} ..... L251,  
     L260, L261, L276, L290, L327, L328  
 \@{\@addfield} ..... L43, L53, L86, L93, L125, L140, L142  
 \@{\@addmarginpar} ..... X331, X1811

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=lt page.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=lt output.dtx, Y=lt hyphen.dtx, Z=lt final.dtx

\@addtobot ..... [X975](#),  
   [X1062](#), [X1129](#), [X1181](#), [X1290](#), [X1349](#)  
 \@addtocurcol ... [X328](#), [X1066](#), [X1999](#)  
 \@addtoblcol ..... [X854](#), [X1562](#)  
 \@addtofilelist .....  
   .... [1005](#), [1028](#), [a104](#), [a106](#),  
   [r64](#), [r132](#), [r187](#), [r531](#), [r653](#), [A749](#),  
   [A752](#), [A759](#), [A762](#), [A769](#), [A772](#),  
   [V169](#), [V188](#), [V196](#), [Z265](#), [Z268](#), [Z629](#)  
 \@addtonextcol .. [X853](#), [X1386](#), [X2000](#)  
 \@addtopreamble ..... [L311](#),  
   [L324](#), [L330](#), [L331](#), [L332](#), [L334](#), [L346](#)  
 \@addtoreset [t16](#), [t39](#), [t44](#), [t89](#), [t115](#), [t118](#)  
 \@addtotopbot .....  
   .... [X1012](#), [X1175](#), [X1343](#), [X1435](#), [X1524](#)  
 \@afterheading ..... [1032](#), [O92](#), [O125](#)  
 \@afterindentfalse ..... [O45](#)  
 \@afterindenttrue .....  
   .... [O43](#), [O124](#), [O208](#), [O231](#)  
 \@alph ..... [104](#), [t139](#), [t152](#), [P399](#)  
 \@ampacol .... [L258](#), [L275](#), [L286](#), [L329](#)  
 \@arabic [t43](#), [t111](#), [t123](#), [t136](#), [t142](#), [P397](#)  
 \@argarraycr ..... [L203](#), [L204](#)  
 \@argdef ..... [f80](#)  
 \@argsrsbox ..... [K469](#)  
 \@argtabularcr ..... [L210](#), [L211](#)  
 \@array ..... [L181](#), [L182](#)  
 \@arrayacol ..... [L168](#), [L258](#)  
 \@arrayclassiv ..... [L169](#), [L331](#)  
 \@arrayclassz ..... [L168](#), [L275](#)  
 \@arraycr ..... [L170](#), [L201](#), [L203](#)  
 \@arrayparboxrestore [K280](#), [K322](#), [L384](#)  
 \@arrayrule .....  
   .... [L309](#), [L311](#), [L315](#), [L317](#), [L319](#), [L346](#)  
 \@arstrut ..... [L192](#), [L237](#), [L343](#)  
 \@arstrutbox .....  
   .... [1009](#), [L185](#), [L218](#), [L343](#), [L385](#)  
 \@author ..... [O8](#), [O32](#)  
 \@auxout [r216](#), [r222](#), [r266](#), [r284](#), [r308](#),  
   [r337](#), [r363](#), [r378](#), [G33](#), [O181](#), [R7](#),  
   [R8](#), [R39](#), [R49](#), [R57](#), [R67](#), [R84](#), [W360](#)  
 \@backslashchar ..... [1011](#), [f219](#),  
   [f469](#), [f582](#), [f594](#), [f598](#), [f599](#), [f604](#),  
   [l234](#), [l236](#), [B266](#), [T1193](#), [T1326](#), [T1415](#)  
 \@badcrerr ..... [1009](#), [l277](#)  
 \@badend ..... [996](#), [1058](#), [l247](#), [H297](#)  
 \@badlinearg ..... [l267](#),  
   .... [M165](#), [M177](#), [M188](#), [M189](#), [M193](#),  
   [M242](#), [M247](#), [M257](#), [M262](#), [M275](#)  
 \@badmath ..... [l251](#), [I262](#),  
   .... [I264](#), [I269](#), [I272](#), [I281](#), [I293](#), [I298](#),  
   [I307](#), [I320](#), [I325](#), [I434](#), [I446](#), [I462](#), [I471](#)  
 \@badpoptabs ..... [l255](#), [L85](#), [L151](#)  
 \@badrequireerror ..... [T434](#), [T1076](#)

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
   f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntr1.dtx,  
   l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
   r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
   w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
   B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
   G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
   M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
   S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
   X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

\@changed@cmd . . . . . 1010, 1021,  
   1022, s3, s63, s223, v131, v267, Z462  
 \@changed@x . . . . . 1021, s3, s211, s219  
 \@changed@x@err . . . . . 1029  
 \@changed@x@mouth . . . . . 1021, s211, s219  
 \@char1b . . . . . r384, r392  
 \@charrb . . . . . r386, r392  
 \@chclass L271, L272, L335, L348, L353  
 \@check@IncludeInRelease . . . . . c68  
 \@check@c . . . . . f189, f191  
 \@check@eq . . . . . f195, f196, f200  
 \@checkcommand . . . . . 1013  
 \@checkend . . . . . 1023, H16,  
   H71, H213, H220, H272, H287, H296  
 \@chnum . . . . .  
   . L279, L298, L335, L350, L351, L352  
 \@circ M622, M640, M649, M654, M657  
 \@circle . . . . . M605, M606  
 \@circlefnt . . . . .  
   M125, M128, M447, M491, M520,  
   M545, M615, M631, M663, M678  
 \@cite . . . . . R36, R95  
 \@cite@ofmt . . . . . R44, R96  
 \@citea . . . . . R35, R37  
 \@citeb . . . . .  
   . R38, R39, R40, R43, R44, R66,  
   R67, R68, R69, R83, R84, R85, R86  
 \@citex . . . . . 783, R20, R21, R29, R34  
 \@citex@checkblank . . . . . R17, R18, R30  
 \@classi . . . . . L271, L307  
 \@classii . . . . . L271, L321  
 \@classiii . . . . . L271, L326  
 \@classiv . . . . . L169, L180, L272  
 \@classoptionslist . . . . .  
   . 1056, T9, T481, T496,  
   T497, T514, T726, T727, T755,  
   T756, T782, T783, T1685, U61, U93  
 \@classv . . . . . L272, L332  
 \@classz . . . . . L168, L179, L271  
 \@cline . . . . . L367  
 \@clnht M195, M196, M204, M206,  
   M208, M218, M225, M273, M672  
 \@clnwd . . . . . M197,  
   M203, M207, M209, M210, M672  
 \@cls@pkg h1356, T290, T291, T304,  
   T305, T865, T875, T923, T970,  
   T1000, T1052, T1061, T1063,  
   T1080, T1538, T1613, T1635, T1665  
 \@clsextension . . . . . 819, 858, T31,  
   T156, T164, T220, T320, T336,  
   T348, T430, T452, T463, T481,  
   T495, T513, T612, T627, T651,  
   T725, T754, T781, T869, T916,  
   T943, T1004, T1017, T1053, U63, U104  
 \@clubpenalty . . . . . r7,  
   r25, r94, r151, J128, J196, O106, O135  
 \@colht . . . . . r22, r91, r148, P273,  
   P275, P278, P284, P285, P298,  
   P299, X114, X231, X242, X251,  
   X252, X387, X399, X434, X447,  
   X474, X505, X535, X541, X545,  
   X555, X560, X645, X704, X777,  
   X815, X859, X884, X903, X943,  
   X965, X1642, X1768, X2130, Z141  
 \@colnum . . . . . P276, X111, X995,  
   X1040, X1109, X1110, X1138,  
   X1146, X1267, X1268, X1300,  
   X1312, X1420, X1421, X1458,  
   X1463, X1507, X1508, X1550,  
   X1557, X1943, X1971, X2010, X2185  
 \@colroom . . . . . 1030, r23, r92, r149,  
   X115, X252, X273, X274, X285,  
   X288, X387, X434, X777, X994,  
   X1039, X1105, X1108, X1137,  
   X1262, X1266, X1299, X1416,  
   X1419, X1502, X1506, X1944,  
   X1972, X2140, X2145, X2190, Z140  
 \@combinedblfloats X750, X2264, X2303  
 \@combinefloats . . . . . X501, X714  
 \@comdblflelt . . . . . X750  
 \@comflelt . . . . . X720, X736, X750  
 \@compatibility . . . . . 1001, 1003, 1004  
 \@cons . . . . .  
   b196, b213, f52, t44, P193, P215,  
   P239, P379, X237, X888, X907,  
   X923, X947, X949, X969, X971,  
   X1141, X1209, X1305, X1378,  
   X1451, X1541, X1644, X1667,  
   X1770, X1795, X1812, X1813, X2191  
 \@contentsline@destination . . . . .  
   . 753, O188, O190, O197  
 \@contfield . . . . . L50, L141, L153  
 \@copy@ . . . . . 97  
 \@copy@DeclareRobustCommand . . . . .  
   . f483, f520, f541, f616, f619  
 \@copy@newcommand . . . . . 91, 98,  
   f306, f484, f520, f562, f591, f616, f622  
 \@copytexsys . . . . . 1003  
 \@ctrerr . . . . . t243, t155, t159, t173, t181  
 \@curfield . . . . . L16,  
   L41, L47, L51, L52, L54, L130, L131  
 \@curline . . . . . L16, L27, L39, L44, L53,  
   L54, L55, L90, L91, L103, L128, L129  
 \@curr@enc . . . . . s154, s156  
 \@curr@file . . . . . 817, 861,  
   863, r226, r227, r236, r238, r262,  
   r270, r397, r416, r551, r566, T850,  
   T1138, T1143, T1149, T1155,

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

T1159, T1170, T1179, T1206,  
 T1269, T1274, T1280, T1303,  
 T1312, T1338, V267, V375, V377  
 \curr@file@reqd .....  
     ..... 861, 863, V267, V377, V381  
 \currbox ..... b275, b276, b277,  
 P60, P91, P95, P129, P160, P164,  
 P193, P214, P215, P239, P257,  
 P259, P261, P319, P322, P327,  
 P331, X213, X214, X225, X226,  
 X228, X229, X237, X311, X312,  
 X853, X854, X1102, X1104, X1112,  
 X1135, X1139, X1141, X1156,  
 X1197, X1209, X1257, X1260,  
 X1297, X1302, X1305, X1322,  
 X1367, X1378, X1410, X1426,  
 X1440, X1451, X1493, X1530,  
 X1541, X1581, X1585, X1596,  
 X1602, X1604, X1608, X1613,  
 X1622, X1631, X1637, X1644,  
 X1667, X1702, X1706, X1718,  
 X1725, X1727, X1731, X1737,  
 X1747, X1762, X1770, X1795,  
 X1813, X1822, X2035, X2036,  
 X2065, X2095, X2100, X2146,  
 X2149, X2161, X2169, X2186, X2191  
 \currdir .....  
     .. 9, a111, a133, a135, a141, a143,  
     a149, a151, a156, a158, a168, a181,  
     a246, a259, a272, T1121, T1143,  
     T1170, T1252, T1274, T1303, T1386  
 \current@cmd ..... 1021, s25, v271  
 \currentcounter .....  
     ..... 1051, 1053, 1059, G39,  
     G40, G42, G53, K379, P471, I352, I489  
 \currentlabel ..... 645, G34,  
 G43, G55, G66, G74, K380, K397,  
 K414, P472, P490, P508, I351, I488  
 \currenvir ..... 287, 298, 1034,  
 l249, H3, H176, H231, H248, H281,  
 H297, J112, K149, T1181, T1193,  
 T1201, T1205, T1212, T1314,  
 T1326, T1334, T1338, T1344,  
 T1403, T1415, T1423, T1427, T1433  
 \currenvinline ..... l249,  
 H177, H232, H249, H282, H298, K150  
 \currext ..... 817, 818, T30,  
 T42, T57, T70, T79, T117, T316,  
 T319, T320, T335, T336, T347,  
 T348, T452, T463, T474, T481,  
 T495, T513, T542, T622, T635,  
 T637, T645, T659, T821, T822,  
 T824, T827, T830, T835, T840,  
 T842, T847, T853, T861, T867,  
 T869, T873, T878, T884, T893,  
 T896, T901, T904, T907, T909,  
 T910, T912, T916, T925, T927,  
 T928, T933, T936, T939, T943,  
 T949, T962, T967, T968, T973,  
 T979, T983, T985, T986, T988,  
 T990, T992, T993, T996, T1002,  
 T1004, T1017, T1030, T1053,  
 T1069, T1070, U28, U63, U73,  
 U78, U81, U104, U106, U109, U161  
 \currextension ..... 997  
 \currlist . P193, P215, P379, X67,  
     X311, X388, X391, X435, X438, X1812  
 \currname . 214, 215, 250, 816–818,  
     842, 1041, 1042, c68, c113, c121,  
     h206, h212, h267, r665, T29, T41,  
     T56, T69, T78, T116, T288, T290,  
     T302, T304, T316, T319, T335,  
     T347, T474, T542, T635, T637,  
     T645, T659, T819, T822, T824,  
     T827, T830, T835, T837, T840,  
     T842, T847, T852, T861, T865,  
     T867, T873, T875, T876, T878,  
     T884, T893, T895, T901, T904,  
     T907, T909, T910, T914, T918,  
     T925, T927, T928, T933, T936,  
     T940, T944, T949, T961, T985,  
     T986, T988, T990, T992, T993,  
     T1030, T1061, T1063, T1070,  
     T1080, T1613, T1635, T1665, U28,  
     U40, U73, U78, U81, U106, U109,  
     U123, U125, U136, U138, U161, U193  
 \currnamestack . 215, 250, 797,  
     799, 869, 1056, h264, T33, T136, V543  
 \curroption ..... 997  
 \curroptions ..... T474,  
     T482, T532, T551, T1070, T1071  
 \currpath . 797, 803, 816, 817, 820,  
     T15, T46, T139, T288, T290, T820,  
     T835, T842, T851, T893, T894, T921  
 \currpkg@reqd .....  
     ... 818, 1053, T318, T845, T847,  
     T856, T889, T903, T906, T921, T923  
 \currsize ..... 996, A639  
 \currtype X119, X878, X879, X880,  
     X881, X898, X899, X900, X901,  
     X1027, X1111, X1121, X1269,  
     X1280, X1422, X1509, X1627,  
     X1752, X2035, X2037, X2038, X2041  
 \curtab ..... 1028,  
     L11, L26, L86, L87, L88, L94, L95,  
     L98, L102, L103, L107, L146, L147  
 \curtabmar ..... L11, L25,  
     L26, L38, L44, L89, L102, L106, L107

**File Key:** a=ltdirname.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr1.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

\@d@r ..... a164, a165  
 \@dashbox .... M324, M325, M326, M327, M328, M331, M337, M339, M349, M351, M352, M353, M354, M358, M362, M365, M382, M384, M385, M386, M387, M390, M394, M396, M405, M407, M408, M409, M410, M413, M417, M422, M674  
 \@dashcnt .... M317, M319, M320, M321, M322, M323, M336, M338, M341, M343, M344, M345, M347, M348, M361, M364, M376, M377, M378, M379, M380, M381, M393, M395, M398, M399, M400, M401, M403, M404, M416, M421, M674  
 \@dashdim .... M316, M317, M318, M319, M320, M322, M325, M327, M328, M329, M336, M338, M340, M341, M342, M343, M344, M347, M351, M353, M354, M355, M363, M366, M375, M376, M377, M378, M380, M384, M386, M387, M388, M393, M395, M397, M398, M399, M400, M403, M407, M409, M410, M411, M419, M424, M674  
 \@date ..... O9, O33  
 \@dbflt ..... 1017, P32, P264  
 \@dblarg .... f693, O54, O142, P12  
 \@dbldeferalist .. 1042, P239, X70, X445, X450, X452, X816, X823, X824, X1752, X1795, X1959, X1988  
 \@dblfloat ..... P31  
 \@dblfloatplacement .....  
     .... r31, r100, r158, P280, X401, X449, X1940, X1968, X2269, X2309  
 \@dblflset ..... P26  
 \@dblfpbot ..... P290, P304, X2353  
 \@dblfpsep ..... P289, P303, X2353  
 \@dblftptop ..... P288, P302, X2353  
 \@dbltoplist .... X69, X232, X235, X237, X397, X398, X445, X446, X755, X759, X761, X762, X1639, X1644, X1764, X1770, X1958, X1986  
 \@dbltopnum .... P283, P297, X107, X127, X238, X240, X766, X1578, X1579, X1643, X1646, X1674, X1679, X1699, X1700, X1769, X1773, X1802, X1807, X1951, X1979  
 \@dbltoproom ..... P284, P286, P298, P300, X108, X1581, X1584, X1585, X1594, X1595, X1598, X1601, X1604, X1608, X1612, X1616, X1621, X1641, X1702, X1705, X1706, X1715,  
 X1716, X1717, X1720, X1724, X1727, X1731, X1736, X1740, X1745, X1746, X1767, X1952, X1980  
 \@dec@text@cmd ..... s3  
 \@declarecommandcopylisthook ... 96, 97, 138, 142, f480, f482, f494, g1018  
 \@declaredoptions ..... 997, T8, T437, T478, T516, T537, T556, T1023  
 \@declareoption ... T435, T436, T444  
 \@defaultfamilyhook ... 1049, 1050, A429, A664, A681, A696, A701, A716  
 \@defaultsubs ..... 1026, v539, v573, v585, H46, H86  
 \@defaultunits .. v214, v218, v219, v220, v235, v328, x179, x181, M13  
 \@defaultunitsset ..... K53, K64, M8, M29, M30, M32, M34, M60, M63, M84, M85, M107, M108, M164, M241, M316, M318, M332, M340, M342, M357, M480, M481, M612, M648, M690, M691, M693, M694, M697, M698, M700, M701, M712, M713, M715, M716, M718, M719, M721, M722  
 \@defdefault@ds ... T435, T440, T445  
 \@deferlist .....  
     ... X68, X384, X393, X394, X397, X402, X404, X410, X431, X440, X442, X778, X786, X787, X798, X803, X804, X1111, X1209, X1269, X1378, X1422, X1451, X1509, X1541, X1627, X1667, X1957, X1985  
 \@definecounter t12, t36, J227, J228, J229, J230, N8, N16, P396, P398, I336  
 \@depth f26, x191, B558, B559, B561, B562, K444, K488, L187, L219, M227, M300, M303, M324, M333, M383, M391, M727, M783, X1851  
 \@dir .... a163, a166, a168, a170, a171  
 \@disable@packageload@do ... T840, V478  
 \@dischyph .....  
     ... 1006, 1045, f730, f764, K289, K310  
 \@doclearpage ..... X296, X371  
 \@documentclass ..... 999  
 \@documentclasshook .....  
     ... 1004, T3, T731, T759, T786  
 \@doendpe H214, H221, H273, H288, J123  
 \@dofilelist 1021, r662, r678, H38, H81  
 \@donoparitem ..... J144, J158  
 \@dot ..... M605, M643  
 \@dotsep ..... O215, O238  
 \@dottedtocline .. O200, O226, O227  
 \@downline ..... M297, M301, M306  
 \@downvector ..... M268, M306

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=lt hyphen.dtx, Z=ltfinal.dtx

\@eha . . . . . c192, f280, f319, f347, l219, l237, l239, l241, l250, l252, l282, r223, r267, r285, s52, s84, v28, v58, v102, v144, v187, v253, v339, x106, z25, z70, z99, z172, z240, z270, z302, z416, z437, z469, z510, z555, z560, z615, z733, z737, z741, z776, z780, z784, z841, z851, z936, z941, z944, z976, z979, z1052, z1055, z1058, z1125, z1131, D146, E24, E81, E915, E924, H175, H230, H247, H280, R71, R88, X1876, X1892  
\@ehb . . . . . l219, l244, l270, l272, l274, X234, X390, X437  
\@ehc . . c181, f128, f155, f469, l219, l277, l280, l286, l288, H509, H524, H539, H552, J220, O31, T1665, I409  
\@ehd . . . . . c99, c156, e88, l219, l246, l254, l257, l259, l265, z118, L100, L109, P6, T692, T923  
\@elt . . . . . 520, 521, 532, 1039, f52, r385, s1572, s1574, t20, t35, t53, t56, A141, A152, A154, A155, A180, A505, A507, A508, A518, A742, B17, B25, X8, X11, X15, X27, X30, X31, X32, X33, X38, X39, X40, X41, X42, X43, X44, X45, X47, X51, X57, X58, X59, X60, X498, X720, X731, X736, X746, X758, X760, X788, X805, X825, X844, X857, X864, X915, X918, X927, X1918, X1930  
\@empty . . . . . 72, 89, 408, 447, 517, 524, 541, 548, 799, 996, 1014, 1016, 1034, 1037, 1038, 1045, k14  
\@emptycol . . . . . 1008, 1035, X198, X245, X248, X277, X281  
\@end@check@IncludeInRelease . . . . . c140, c142  
\@end@tempboxa . . . . . K36, K45, K208, K279, K476, K486  
\@enddocument@kernel@warnings . . . . . H39, H41, H101  
\@enddocumenthook . H70, T1025, T1047  
\@endfloatbox P190, P211, P236, P248  
\@endparenv . . . . . 660, J120, J123  
\@endparpenalty . . . . . 652, p15, J23, J124, I438, I450, I476  
\@endpbox . . . . . L193, L236, L266, L333, L384, L387  
\@endpfalse . . . . . 1000, H181, H235, H252, H285, J129, J131, J135, J136, J138, K152  
\@endpeltrue . . . . . J138  
\@endptrue . . . . . J124, J126, J134  
\@endpreamblehook . . . . . 334  
\@endtheorem . . . . . N13, N19, N25, N35  
\@enlargepage . . . . . X1861, X1866, X1868  
\@ensuredmath . . . . . I420, I422  
\@enumctr . . . . . J234, J237, J238  
\@enumdepth . . . . . 663, J226, J232, J233, J234  
\@enumspacing . . . . . 663  
\@eqcnt . . . . .  
\@eqnfalse . . . . . I374  
\@eqntrue . . . . . I346, I353, I411, I490  
\@eqopen . . . . . I344, I378, I380, I391, I401  
\@er@ext . . . . . 998  
\@err@ . l37, l41, l44, l52, l64, l68, l71, l79  
\@esphack . . . . . 995, p38, p131, p345, p362, p379, p396, G35, P385, Q17, Q19, Q34, R74, R90, X1884  
\@evenfoot . . . . . S12, S15, X613, X672  
\@evenhead . . . . . S12, S15, X612, X671  
\@execute@begin@hook . . . . . 622, H178, H182, H185  
\@executeoption . . . . . 998  
\@expandtwoargs . . . . . c185, f217, T222, T287, T480, T516, T570, T579  
\@expast . . . . . L239, L267  
\@expl@@@filehook@clear@replacement@flag@@ T386, T409, V296, V319, V340, V516  
\@expl@@@filehook@drop@extension@CN . . . . . 861, V293, V294, V316, V317, V337, V338, V518  
\@expl@@@filehook@file@pop@@ . . . . . T858, V157, V522, V532  
\@expl@@@filehook@file@pop@assign@@nnnn . . . . . 858, V162, V524  
\@expl@@@filehook@file@push@@ . . . . . T841, V151, V520  
\@expl@@@filehook@if@file@replaced@CTF . . . . . 861, 863, V290, V313, V334, V514  
\@expl@@@filehook@if@no@extension@nTF V286, V309, V330, V504, V506, V531  
\@expl@@@filehook@normalize@file@name@@w . . . . . V292, V315, V336, V512  
\@expl@@@filehook@resolve@file@subst@@w T384, T407, V289, V312, V333, V510  
\@expl@@@filehook@set@curr@file@@NN . . . . . 816, 861, 863, T383, T406, T813, V375, V381, V508  
\@expl@@@hook@curr@name@pop@@ . . . . . h1524, T89  
\@expl@@@hook@curr@name@push@on 1054

**File Key:** a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\@expl@@@initialize@call@@ [h1524](#), [H186](#)  
 \@expl@@@shipout@add@background@box@@n  
     ..... [W406](#), [W491](#)  
 \@expl@@@shipout@add@background@picture@@n  
     ..... [W406](#), [W495](#)  
 \@expl@@@shipout@add@firstpage@material@@N@fileswfalse  
     ..... [W406](#), [W488](#)  
 \@expl@@@shipout@add@foreground@box@@n  
     ..... [W406](#), [W493](#)  
 \@expl@@@shipout@add@foreground@picture@@n  
     ..... [W406](#), [W497](#)  
 \@expl@char@generate@@nn ..  
     ..... [1053](#), [e142](#), [f709](#)  
 \@expl@cs@(thing)@spec@@N .. [1052](#)  
 \@expl@cs@argument@spec@@N ..  
     ..... [e139](#), [e150](#), [f610](#)  
 \@expl@cs@prefix@spec@@N ..  
     ..... [e138](#), [e149](#), [f607](#)  
 \@expl@cs@replacement@spec@@N ..  
     ..... [e140](#), [e151](#), [f568](#), [f602](#), [f610](#)  
 \@expl@cs@to@str@@N ..  
     ..... [1052](#), [e133](#), [e136](#),  
           [e145](#), [e147](#), [f477](#), [f534](#), [f554](#), [f556](#),  
           [f557](#), [f570](#), [f582](#), [f594](#), [f598](#), [f599](#), [f604](#)  
 \@expl@finalise@setup@@ ..  
     ..... [e30](#), [Z254](#), [Z255](#)  
 \@expl@pop@filename@@ ..  
     .... [1052](#), [e26](#), [T88](#), [T92](#), [T98](#), [T109](#)  
 \@expl@push@filename@@ ..  
     ..... [797](#), [1052](#), [e24](#), [T37](#),  
           [T39](#), [T52](#), [T54](#), [T64](#), [T76](#), [T96](#), [T102](#)  
 \@expl@push@filename@aux@@ ..  
     ..... [250](#), [1052](#), [e25](#),  
           [h1434](#), [T37](#), [T48](#), [T60](#), [T64](#), [T82](#), [T102](#)  
 \@expl@str@if@eq@@nnTF ..  
     ... [1052](#), [e137](#), [e148](#), [f441](#), [f443](#), [T287](#)  
 \@expl@str@map@function@NN ..  
     ..... [104](#), [e141](#), [e152](#), [f706](#)  
 \@expl@str@map@function@NN and  
     \@expl@char@generate@@nn ... [1053](#)  
 \@expl@sys@load@backend@@ [e21](#), [e23](#), [r17](#)  
 \@expl@text@lowercase@@n .. [Z551](#)  
 \@expl@text@uppercase@@n .. [1060](#), [Z551](#)  
 \@extra@page@added .. [H57](#), [W376](#)  
 \@failedlist .. [X842](#), [X865](#), [X881](#),  
     [X888](#), [X901](#), [X907](#), [X923](#), [X937](#), [X960](#)  
 \@fcolmadefalse .. [X833](#)  
 \@fcolmadetrue .. [X921](#)  
 \@file-subst@<file> .. [859](#)  
 \@filef@und .. [857](#), [1047](#), [1049](#),  
     [1054](#), [r436](#), [r457](#), [r478](#), [r501](#), [r518](#),  
     [r531](#), [r584](#), [V155](#), [V172](#), [V188](#), [V196](#)  
 \@filehook@file@push .. [817](#)

\@filehook@set@CurrentFile ..  
     ..... [r314](#), [T843](#), [V152](#), [V364](#)  
 \@filelist ..  
     ..... [r63](#), [r131](#), [r186](#), [r652](#), [r653](#), [r664](#),  
           [A749](#), [A759](#), [A769](#), [Z265](#), [Z613](#), [Z629](#)  
 \@fileswfalse ..  
     ..... [r199](#), [T1117](#), [T1118](#), [T1248](#), [T1249](#)  
 \@fileswith@pti@ns .. [997](#), [T434](#), [T540](#),  
     [T559](#), [T717](#), [T718](#), [T722](#), [T724](#),  
     [T752](#), [T753](#), [T779](#), [T780](#), [T807](#), [T1014](#)  
 \@fileswith@ptions ..  
     ..... [T712](#), [T713](#), [T715](#), [T719](#)  
 \@fileswithoptions ..  
     ..... [T612](#), [T619](#), [T627](#), [T710](#)  
 \@filestrue ..  
     ..... [r5](#), [T1100](#), [T1103](#), [T1156](#),  
           [T1160](#), [T1231](#), [T1234](#), [T1289](#), [T1293](#)  
 \@finalstrut ..  
     ..... [1009](#)–[1011](#), [K384](#), [K401](#),  
           [K418](#), [K487](#), [L385](#), [P477](#), [P495](#), [P513](#)  
 \@firststampfalse .. [L254](#), [L277](#), [L294](#)  
 \@firststamptrue .. [L262](#)  
 \@firstcolfirstmark ..  
     ..... [X2246](#), [X2247](#), [X2251](#)  
 \@firstcoltopmark .. [X2244](#), [X2252](#)  
 \@firstcolumnfalse .. [X2236](#), [X2281](#)  
 \@firstcolumntrue .. [r28](#),  
     [r97](#), [r155](#), [X98](#), [X207](#), [X2255](#), [X2287](#)  
 \@firstofone ..  
     .. [96](#), [183](#), [1023](#), [1052](#), [d12](#), [d100](#),  
           [d108](#), [d166](#), [e34](#), [e61](#), [e76](#), [f212](#),  
           [f465](#), [f470](#), [f471](#), [f474](#), [f475](#), [r125](#),  
           [r180](#), [r404](#), [s68](#), [s153](#), [x346](#), [z53](#),  
           [z81](#), [z146](#), [z221](#), [z251](#), [z282](#), [z886](#),  
           [H69](#), [L372](#), [P10](#), [R38](#), [R66](#), [R83](#),  
           [T859](#), [T888](#), [V353](#), [V370](#), [Z345](#), [I418](#)  
 \@firstoftwo ..  
     .. [344](#), [346](#), [a90](#), [c82](#), [e35](#), [e43](#), [f212](#),  
           [f442](#), [f444](#), [f466](#), [f537](#), [f587](#), [f645](#),  
           [f655](#), [f665](#), [f692](#), [f716](#), [r458](#), [r502](#),  
           [r519](#), [s133](#), [t191](#), [t196](#), [z890](#), [A510](#),  
           [E69](#), [E836](#), [E887](#), [E903](#), [G19](#), [S16](#),  
           [T161](#), [T193](#), [T205](#), [T228](#), [T246](#), [T1674](#)  
 \@firstab ..  
     .. [1009](#), [L2](#), [L74](#), [L75](#), [L76](#), [L106](#), [L118](#)  
 \@flcheckspace .. [X989](#), [X1025](#), [X2136](#)  
 \@flfail ..  
     .. [X865](#), [X916](#), [X937](#), [X947](#), [X960](#), [X969](#)  
 \@float .. [P26](#), [P32](#)  
 \@floatboxreset .. [P101](#), [P170](#), [P174](#)  
 \@floatpenalty ..  
     .. [P3](#), [P53](#), [P55](#), [P58](#), [P122](#), [P124](#),  
           [P127](#), [P191](#), [P194](#), [P199](#), [P201](#),

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspac.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=lt hyphen.dtx, Z=ltfinal.dtx

P212, P216, P221, P223, P237,  
 P241, P311, P313, P317, P321, P379  
 \@floatplacement .....  
     . 999, r31, r100, r158, P271, X149,  
     X209, X253, X477, X1941, X1969  
 \@flsetnum .....  
     . X986, X1022, X1109, X1267,  
     X1420, X1507, X1578, X1699, X2104  
 \@flsettextmin .....  
     X1085, X1237, X1406, X1489, X2120  
 \@flstop ..... X2006  
 \@flsucceed .....  
     . X858, X866, X915, X949, X971  
 \@fltovf ..... l273, P93, P162, P322  
 \@flupdates ..... X992, X1037, X2182  
 \@flushglue ..... j17,  
     H359, H363, H369, H379, H382,  
     H387, H420, H442, J76, K296, K317  
 \@fnssymbol ..... t141, t160  
 \@font@aliasinfo ..... x539  
 \@font@info .....  
     v133, v171, v177, v366, v383, v621,  
     w478, x30, x38, x46, x74, x87, x154,  
     x200, x214, x225, x239, x255, x261,  
     x274, x281, x288, x293, x303, x315,  
     x327, x491, x503, x508, x515, x545,  
     x558, x566, z180, z197, z206, z312,  
     z327, z374, z420, z489, z495, z539,  
     z552, z635, z724, z767, z832, z926,  
     z1093, z1122, E55, E56, E99, Z464  
 \@font@series@contextfalse .....  
     . A502, A538  
 \@font@series@contexttrue .....  
     . A521, A525, A537  
 \@font@shape@subst@warning . 452,  
     w445, w448, w453, w511, w658, w661  
 \@font@warning v3, v535, v540, v567,  
     v574, w456, x19, x33, x41, x49,  
     x61, x77, x476, x490, x502, x507,  
     x514, x557, x565, y30, H43, H83, Z286  
 \@fontenc@load@list .....  
     . 1050, s1573, A742, B17, B25  
 \@fontswitch ..... 1009, D126, D128  
 \@footnotemark .....  
     P454, P460, P522, P528, P529, P555  
 \@footnotetext .....  
     K349, P454, P460, P461, P538, P544  
 \@for ..... 783,  
     k16, r232, r304, r359, r664, R36,  
     R65, R82, T230, T247, T478, T497,  
     T514, T532, T537, T551, T556,  
     T592, T602, T1071, T1108, T1239  
 \@forced@seriesfalse w394, w407, x140  
 \@forced@seriestrue .... w398, w409

\@forloop ..... k19, k20  
 \@fornoop ..... k15, k23, k29  
 \@fortmp ..... k17, k18, k26, T590,  
     T592, T1107, T1108, T1238, T1239  
 \@fpbot ..... P290, P304, X863, X2347  
 \@fpmin .. P278, P287, P301, X113,  
     X920, X1949, X1977, X2199, X2216  
 \@fps ..... P41, P42,  
     P44, P47, P64, P110, P111, P113,  
     P116, P133, X2027, X2029, X2032  
 \@fpsaddrdefault .....  
     . P45, P48, P114, P117, X2024  
 \@fpsep ..... P289,  
     P303, X861, X870, X942, X964, X2347  
 \@fpstype .....  
     . X983, X1004, X1005, X1019,  
     X1050, X1051, X1075, X1077,  
     X1080, X1082, X1133, X1189,  
     X1190, X1225, X1228, X1231,  
     X1234, X1295, X1357, X1358,  
     X1396, X1398, X1401, X1403,  
     X1477, X1480, X1483, X1486,  
     X1575, X1590, X1592, X1610,  
     X1619, X1655, X1656, X1696,  
     X1711, X1713, X1733, X1743,  
     X1782, X1783, X2020, X2036,  
     X2038, X2040, X2043, X2044,  
     X2045, X2047, X2048, X2052,  
     X2053, X2055, X2056, X2090,  
     X2092, X2094, X2106, X2108,  
     X2122, X2124, X2154, X2157, X2168  
 \@ftp ..... P288, P302, X860, X2347  
 \@frameb0x 1015, 1016, K179, K207, K209  
 \@framebox ..... K186, K193, K197  
 \@framepicbox ..... K186, K193, K230  
 \@freelist ..... 1043, b196, b213,  
     b275, P60, P129, P319, P320, X29,  
     X34, X48, X56, X213, X499, X732,  
     X747, X761, X866, X1812, X1813  
 \@generic@error ..... 1014  
 \@generic@message ..... 1014  
 \@getcirc ..... M437,  
     M485, M514, M541, M613, M629  
 \@getfpsbit .....  
     . X980, X1016, X1572, X1693, X2063  
 \@getlarrow ..... M266, M274, M276  
 \@getlinechar ..... M190, M229  
 \@getpen ..... p34, p37, p46, p117  
 \@getrarrow ..... M267, M274, M283  
 \@glossaryfile ..... Q21, Q22, Q31  
 \@newline ..... p95, p101, p108, p111  
 \@gobble .....  
     . 532, 817, 1010, 1011, 1022, 1028,  
     1052, c160, c184, d11, d98, e33, e65,

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

e88, f111, f133, f208, f219, f236, f240, f277, f283, f286, f296, f316, f322, f325, f334, f344, f350, f353, f362, f380, f384, f386, f387, f389, f397, f401, f403, k6, k9, l101, l127, l147, l155, l180, l189, l202, o14, p75, p483, r64, r132, r187, r411, r413, r652, s29, v536, v569, x345, y26, z28, z30, z378, z389, z453, z500, z501, z530, z536, z544, z549, z567, z581, z591, z600, z613, z630, z639, z713, z715, z719, z727, z761, z770, z822, z824, z835, z919, z929, z1010, z1015, z1084, z1115, A139, A180, A518, A752, A762, A772, E864, H239, O143, O144, O145, O146, O147, O182, P7, R11, R45, R46, T680, T838, T1092, T1152, T1179, T1283, T1312, T1396, T1401, T1475, T1619, T1631, V261, X619, X620, X621, X678, X679, X680, X927, X1932, X2200, X2217, Z268, Z397, Z558, Z564, Z629  
 \@gobble@AddToHook@args h1540, h1541  
 \@gobble@IncludeInRelease ..... c68  
 \@gobble@RemoveFromHook@arg ..... h1543, h1544  
 \@gobblecr ..... p481, p482  
 \@gobblefour ..... f208, z24, z375, z491, z493, z497, z499, z509, z513, z637, z689, T1181, T1314, T1403  
 \@gobblethree ..... 799, 1021, f208, f596, f609, T128  
 \@gobbletwo ..... 1026, 1034, e123, e128, f175, f176, f208, k12, r32, r101, r159, v541, v575, z132, H23, H44, H76, H84, S11, S13, T1151, T1282, T1395, V260, Z292  
 \@gttempa ..... 1005, f126, f127, f181, f183, f528, f536, r596, r597, r603, r604, r605, r630, r631, r633, r634, r635, L3, L5, L6, L7, L8, T286, T288, T301, T302, T321, T323, T337, T339, T349, T351  
 \@halfwidth ..... M2, M126, M129, M131, M227, M299, M302, M324, M333, M349, M361, M364, M383, M391, M405, M416, M421, M680, M706, M725, M726, M727, M766, M781, M782, M783  
 \@halignto .... L170, L174, L177, L191  
 \@hangfrom ..... O66, O117, O138  
 \@height ... b469, f26, p350, p358, p384, p392, s293, s295, x190, B340, B558, B559, B561, B562, K163, K168, K216, K226, K444, K488, L186, L219, L359, L376, M227, M300, M303, M324, M333, M351, M359, M383, M391, M407, M414, M590, M600, M726, M782, X1851  
 \@highpenalty ..... p118, Z3  
 \@heightab .... 1028, L11, L21, L23, L74, L86, L95, L96, L111, L146, L147  
 \@hline .... M167, M179, M226, M265  
 \@holdpg ..... X122, X300, X302, X303, X308, X309, X310  
 \@hspace ..... 996, p437, p438, p453  
 \@hspacer ..... 996, p437, p452  
 \@hvector ..... M244, M259, M265  
 \@ialph ..... 1011  
 \@icentercr ..... H337, H338  
 \@iden ..... f215  
 \@if ..... f171, f172, f174  
 \@if... ..... 1011  
 \@if@DeclareRobustCommand ..... 98, f483, f507, f518, f519, f523, f614, f615, f618, i125  
 \@if@newcommand ..... 91, 98, 100, 262, 1057, f304, f484, f508, f519, f561, f573, f578, i126, i137  
 \@if@newlist ..... 1038  
 \@if@pti@ns ..... T222, T225, T227, T244, T245  
 \@if@ptions ..... T219, T220, T221, T873, T968  
 \@if@short@command ..... 1011  
 \@ifatmargin ..... L55, L106  
 \@ifbothcounters ..... t61, t72, t80, t88, t104, t106, t115, t117  
 \@ifclasslater .... T163, T171, T178  
 \@ifclassloaded ... T155, T266, T275  
 \@ifclasswith .... T219, T268, T277  
 \@ifdefinable ..... 999, 1011, f84, f86, f130, f132, f238, s14, s17, t11, u3, A618, K115, N7, N15, N22  
 \@iffileonpath ..... r432, r453, r474, r491, r512  
 \@ifl@aded ..... 817, T155, T156, T157, T822, T949, T967  
 \@ifl@t@r .... s1538, T169, T176, T182, T186, T188, T199, T200, T701  
 \@ifl@ter ..... s1577, s1578, T163, T164, T181, T861, T996  
 \@ifl@ter@Q ..... s1577, s1578  
 \@ifnch .... e49, e50, f677, f679, f691  
 \@ifnextchar ..... 996, 1009, 1039, a101, e45, f673, f678, f692, p93, p482, r544, t13,

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspacel.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

x411, H336, J143, K9, K11, K18,  
 K20, K26, K47, K121, K122, K128,  
 K129, K136, K140, K185, K186,  
 K192, K193, K198, K231, K239,  
 K247, K254, K258, K327, K331,  
 K335, K428, K433, K456, K463,  
 K468, L57, L181, L203, L210, M23,  
 M132, M143, M453, N3, N5, N28,  
 P27, P264, P324, P452, P519,  
 P536, R3, R17, R29, T293, T307,  
 T696, T711, T716, T1101, T1104,  
 T1232, T1235, X209, X2008, I342  
 \@iforloop ..... k21, k22  
 \@ifpackagelater .....  
     794, 1040, T163, T170, T177  
 \@ifpackageloaded .....  
     794, T155, T265, T274, X1992  
 \@ifpackagewith 794, T219, T267, T276  
 \@iframebox ..... K199, K200, K201  
 \@iframedpicbox ..... K231, K232  
 \@ifstar ..... f73, f692, p59,  
     p71, p330, p437, t102, t113, v206,  
     y121, H325, H332, H563, H572,  
     L56, L202, L209, M142, M605,  
     O52, O142, T435, T475, X1856, I377  
 \@ifundefin@d@i . f635, f636, f653, f656  
 \@ifundefin@d@ii ..... f635, f638, f641  
 \@ifundefined ..... 1009,  
     e132, f127, f134, f154, f161, f183,  
     f194, f277, f283, f316, f322, f344,  
     f350, f380, f397, f478, f630, g2664,  
     t3, t7, t16, t50, t62, t64, v100,  
     v186, x424, z410, E54, E866, G23,  
     H141, H158, H174, H229, H246,  
     H279, N21, R40, R68, R85, S3,  
     S7, T128, T153, T388, T394, T411,  
     T423, T499, T533, T552, T824, V485  
 \@ignore... ..... 1031, 1032  
 \@ignorefalse .....  
     995, H4, H180, H216, H223,  
     H234, H251, H274, H284, H289, P384  
 \@ignoretrue ..... 995, 1031, p200,  
     p213, H4, H7, I335, I338, I371, I514  
 \@iiiminipage .....  
     K329, K333, K336, K337, K338  
 \@iiiparbox ..... K241,  
     K249, K256, K259, K260, K261, K366  
 \@iiminipage ..... K332, K334  
 \@iinput ..... 346, r544, r545  
 \@iiparbox ..... K255, K257  
 \@iirsbox ..... K468, K477  
 \@imakebox ..... K26, K41, K138  
 \@imakedpicbox . K47, K48, K143, K233  
 \@iminipage ..... K328, K330

**File Key:** a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

```

\@kernel@after@begindocument ...
 1056, e7, i60, r58, r75, U173
\@kernel@after@begindocument@before
 334, r16, r75, w731
\@kernel@after@enddocument ...
 e1, H15, W350
\@kernel@after@enddocument@afterlastpage
 e1, h293, H19, W356
\@kernel@after@para@after 296, n8, n64
\@kernel@after@para@end ...
 295, 296, n8, n57
\@kernel@after@shipout@background
 880, 1055, W72, W158
\@kernel@after@shipout@lastpage .
 892, W114, W119, W158, W367, W372
\@kernel@before@{hook} ...
 198
\@kernel@before@begindocument ...
 1056, e7, r56, r75, W396
\@kernel@before@enddocument ...
 618, H13, H105
\@kernel@before@para@before ...
 295, 296, n8, n19
\@kernel@before@para@begin ...
 295, 296, n8, n27
\@kernel@before@shipout@background
 880, W68, W70, W158
\@kernel@currpathstack ...
 799, 1056, T45, T47, T91, T123
\@kernel@make@file@csname . V288,
 V291, V311, V314, V332, V335, V364
\@kernel@rename@newcommand ...
 91, 93, f285, f302, f339, f367, f372, f385
\@killglue ...
 M59, M73, M103, M115, M123
\@kludgeins ...
 X319, X320, X321, X323, X376,
 X377, X423, X424, X502, X518,
 X519, X525, X526, X527, X536,
 X552, X556, X566, X1852, X1883
\@labels ...
 651, J27, J146, J147, J189, J206, J207
\@largefloatcheck ...
 1009, P192, P213, P238, P256
\@lastchclass ...
 .. L262, L272, L273, L275, L283,
 L308, L322, L326, L335, L348, L349
\@latex@error ...
 c97, c156, c181, c192, e80, f128, f155,
 f278, f317, f345, f469, l163, l217,
 l233, l239, l241, l244, l246, l248,
 l252, l254, l256, l259, l263, l268,
 l272, l274, l276, l277, l279, l282,
 l286, l288, o9, r223, r267, r285, s50,
 s84, v5, v28, v58, v102, v144, v187,
 v253, v339, x105, y100, y111, z23,
 z68, z97, z117, z170, z238, z268,
 z300, z323, z339, z416, z437, z469,
 z509, z513, z555, z560, z615, z683,
 z689, z733, z737, z741, z776, z780,
 z784, z841, z851, z936, z941, z944,
 z976, z979, z1052, z1055, z1058,
 z1125, z1131, A49, A60, A82, A93,
 A600, A720, D143, H175, H230,
 H247, H280, H509, H524, H539,
 H551, J219, L100, L109, O31, P6,
 P83, R71, R88, T623, T674, T687,
 T874, T922, T969, T1060, T1077,
 T1085, T1090, T1112, T1165,
 T1243, T1298, T1634, T1664,
 X234, X390, X1874, X1891, I409
\@latex@info ...
 285, f224, f298,
 f336, f364, f730, l163, l208, s85, E76
\@latex@info@no@line ...
 1013, l163, l209, W89, X576
\@latex@note ...
 l190, v33
\@latex@note@no@line ...
 1057, l190, T1142, T1148
\@latex@warning ...
 l163, l215,
 r339, s55, G14, M449, P260, R42,
 R69, R86, T1204, T1211, T1337,
 T1343, T1426, T1432, W172, X2030
\@latex@warning@no@line ...
 e98,
 f202, l163, l216, r19, r88, r145, r650,
 G8, G26, G27, H51, H58, H91, O32,
 T289, T303, T702, T862, T997,
 T1169, T1273, T1279, T1302,
 T1385, T1392, T1459, T1536, W79,
 W98, X243, X275, X1827, X2096
\@latexbug ...
 l275, X333, X1813
\@latexerr ...
 1010, l215, X437
\@latexinfo ...
 1008
\@latexrelease@catcode@null g6, g2665
\@libbibitem ...
 R3, R4
\@ldots ...
 B504, B506
\@leftcolumn ...
 X121, X2237, X2258, X2282, X2291
\@leftmark ...
 S16, S50
\@let@token ...
 1009, 1020, 1027, e49,
 e51, f677, f680, f683, f691, p410,
 p411, p418, D83, D96, I243, I245, I248
\@align ...
 I195, I197
\@linechar ...
 1039, M190, M191, M192, M196,
 M197, M199, M204, M206, M207,
 M208, M209, M211, M215, M216,
 M219, M220, M225, M272, M670
\@linenfnt ...
 M124, M127, M190,
 M265, M273, M304, M307, M677

```

**File Key:** a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=lptra.dtx, o=ltmeta.dtx, p=ltspac.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=lttoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfnctcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\@linelen . . . . . M164, M165, M176, M177, M203, M210, M219, M221, M226, M227, M228, M241, M242, M256, M257, M300, M303, M305, M306, M671  
 \@list . . . . . 652  
 \@listctr . . . . . 651, 995, J202, J225, R9  
 \@listdepth . . . . . . 651, J23, J35, J38, J43, J99, K350  
 \@listfiles . r62, r130, r185, r656, r671  
 \@listi . . . . . 652  
 \@listii . . . . . 652  
 \@listvi . . . . . 652  
 \@lnbk . . . . . 1033  
 \@loadwithoptions . . . . . . T629, T651, T661, T670  
 \@lowpenalty . . . . . p117, Z3  
 \@ltab . . . . . L71, L106  
 \@ltxnomath . . . . . 1008  
 \@m . . . . . 273, 1025, b21, b396, b398, b399, b465, b466, p288, p429, p434, r45, r114, r172, J80, M213, M217, R37  
 \@mainaux . . . . . 1019, r3, r37, r38, r106, r107, r164, r165, r216, r298, r337, r353, r378, H22, H75  
 \@makebox . . . . . K11, K20, K25  
 \@makecaption . . . . . P24  
 \@makecol . . . . . X261, X413, X460, X480  
 \@makefcolumn . . . . . X393, X394, X402, X404, X440, X442, X450, X452, X2195, X2197, X2213, X2214  
 \@makefnmark . . . . . P400, P532  
 \@makefntext . . . . . . K383, K400, K417, P476, P494, P512  
 \@makeother . a79, a100, a129, f695, f696, v406, v407, v408, v409, v410, v411, v412, v413, v414, v415, v416, H431, H452, H545, H560, H570, T375, T376, T1186, T1319, T1408  
 \@makepicbox . . . . . . K10, K19, K46, M366, M424  
 \@makespecialcolbox . 998, X503, X522  
 \@marbox . . . . . P320, P322, P326, P330, P331, P379, X1812, X1822, X1825, X1833, X1835, X1836, X1838, X1839, X1840, X1849  
 \@marginparreset . . . P343, P361, P370  
 \@markright . . . . . S33, S48  
 \@maxdepth . . . . . 1005, 1026, r60, r128, r183, X91, X486, X514, Z138  
 \@maxtab . . . . . 1009, L2, L94  
 \@medpenalty . . . . . p118, Z3  
 \@meta@family@list . . . . . . A116, A141, A153, A242, A506  
 \@midlist . X66, X499, X500, X1027, X1029, X1141, X1305, X1956, X1984  
 \@minipage . . . . . 1031  
 \@minipagefalse . . . . . J181, K323, K325, K363, P187, P250, P345, P363  
 \@minipagerestore . . . . . K351, K353  
 \@minipagetrue . . . . . K324, P186  
 \@minus . . . . . f26, X2340, X2341, X2342, X2345, X2346  
 \@missing@onefilewithoptions . . . . . . T836, T892, T1010  
 \@missingfile@area . . . . . . r557, r598, r611, T894  
 \@missingfile@base . . . . . . r557, r599, r612, T895  
 \@missingfile@ext . r557, r600, r613, T896  
 \@missingfileerror . . . . . . 346, 347, 817, 1017, r552, r567, r577, r586, T893, T990  
 \@mkboth . . . . . S11, S13  
 \@mklab . . . . . J45, J140  
 \@mkpream . 1018, 1032, L189, L234, L262  
 \@mparbottom . . . . . . P387, P388, X118, X476, X1823, X1831, X1832, X1833, X1834  
 \@mpargs . . . . . K342, K366  
 \@mparswitchfalse . . . . . X102  
 \@mpfn . K348, P452, P457, P541, P545  
 \@mpfootins K357, K358, K361, K367, K374, K375, K392, K393, K409, K410  
 \@mpfootnotetext . . . . . K349, K369  
 \@mplistdepth . . . . . K350, K367  
 \@multicnt . 1027, L370, L372, L373, L374, L381, L382, L383, M103, M104, M106, M115, M116, M118, M667, M704, M706, M707, M708, M710, M711, M717, M723, M734, M738, M764, M766, M768, M770, M771, M775, M779, M790, M794  
 \@multiplelabels . . . . . r33, r102, r160, G25, G31, H49, H55, H89, H95  
 \@multiput . . . . . M86, M95, M98  
 \@multispan . . . . . L371, L375, L379  
 \@mypkg@name . . . . . 841  
 \@mypkg@other@name . . . . . 841  
 \@namedef . . . . . 997, 1013, f50, r391, v135, v136, v160, w5, w525, x418, z360, E57, E93, E869, G28, H208, H218, H245, H271, H483, H492, L175, N12, N13, N18, N19, N23, N24, N25, T827, T1103, T1234, V483, Z466, Z467, I413, I414  
 \@nameuse . . . . . 1013, f51, r335, r376, r390, N23, S5, V486, X607, X665

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\@nbitem ..... J168, [J221](#)  
 \@ne ..... [273](#), [1013](#), [b16](#)  
 \@needsf@rmat ..... T697, T700, T705  
 \@needsformat ..... T685, T695, T699  
 \@negargfalse ..... M186  
 \@negargtrue ..... M185  
 \@newcommand ..... f79, [f80](#)  
 \@newctr ..... t13, [t15](#), N8  
 \@newenv ..... f150, f151, [f160](#)  
 \@newenva ..... f148, [f149](#)  
 \@newenvb ..... f150, [f151](#)  
 \@newfontswitch ..... [1002](#), [1007](#)  
 \@newl@bel .. [1025](#), [G22](#), H24, H77, R10  
 \@newline ..... p94, [p96](#)  
 \@newlistfalse .....  
     ... [J29](#), [J33](#), J108, J182, X600, X658  
 \@newlisttrue ..... [J29](#), [J33](#), J87  
 \@next b275, P60, P129, P319, P320,  
     X9, X213, X311, X877, X897, X1812  
 \@nextchar L269, L270, L330, L331, L332  
 \@nil .... a164, a165, c13, c19, c80,  
     c101, c102, c114, c115, c162, c163,  
     c164, f53, f54, f58, f63, f135, f439,  
     f440, f585, f671, f672, k13, k19, k27,  
     q14, r246, r247, r248, s89, s110,  
     s996, s1000, s1063, s1075, s1077,  
     v358, v369, v485, v500, v604, v607,  
     v608, v616, w431, w433, w465,  
     w467, w645, w647, w671, w673,  
     x350, x351, x353, x366, x372, x376,  
     x377, x413, x434, x439, x519, x533,  
     y26, y44, y53, y57, z40, z479,  
     z487, z520, z1136, z1138, D58, D62,  
     L367, L368, T90, T91, T97, T105,  
     T108, T115, T140, T189, T190,  
     T201, T202, T212, T213, T215,  
     T384, T407, T450, T455, T571,  
     T591, T595, T601, T605, T793,  
     T802, T816, T817, T1509, T1514,  
     T1517, T1519, T1520, T1535,  
     T1555, T1572, T1626, T1648,  
     T1672, V168, V176, V370, V412, V414  
 \@nmbrlistfalse ... 995, [J33](#), J46, J91  
 \@nmbrlisttrue ..... J225  
 \@nnil f417, f423, [k13](#), k20, k21, k22,  
     k28, r247, r248, v214, v218, v219,  
     v220, v235, x179, x181, x345, x347,  
     x359, x361, x366, x380, x382, x389,  
     x400, x401, x403, x434, x439, M13,  
     T738, T739, T746, T766, T767, T774  
 \@no@font@optfalse ..... y17, [y129](#)  
 \@no@lnbk ..... 1033, p9, p10, [p40](#)  
 \@no@pgbk ..... 1026, p7, p8, [p32](#)  
 \@nobreak... ..... 1031, [1032](#)

\@nobreakfalse ..... 1032,  
     p120, p122, J193, O94, O129, O157,  
     P182, X165, X192, X1150, X1316  
 \@nobreaktrue ..... p121, O126, P181  
 \@nocnterr ..... 1006, [1007](#), [l240](#)  
 \@nocounterr .....  
     ... 1007, [l240](#), t4, t8, t16, t62, t64, N21  
 \@nodocument ..... 1032, [1033](#),  
     [l245](#), n101, r68, r136, r191, r295,  
     H166, P39, P108, X156, X183, X212  
 \@noitemargfalse ..... [J32](#), J200  
 \@noitemargtrue ..... [J32](#), J143  
 \@noitemerr ..... 1278,  
     p248, p266, p303, p326, J69, J81, J107  
 \@noitemerror ..... 1019  
 \@noligs ... 997, [1000](#), [1027](#), [1030](#),  
     1031, H432, H453, H561, H571, [H582](#)  
 \@nolnbk ..... 1033  
 \@nolnerr .....  
     ... 1009, [l238](#), p42, p113, H324, H331  
 \@nomath ..... 993, [v1](#), v337,  
     A551, A586, A592, A613, A615, A637  
 \@noparitemfalse ..... [J30](#), J145  
 \@noparitemtrue ..... [J30](#), J66  
 \@noparlistfalse ..... [J31](#), J70  
 \@noparlisttrue ..... [J31](#), J67  
 \@nopgbk ..... 1026  
 \@normalcr ..... 1048, [p52](#), p92, K322  
 \@normalsize .....  
     ... 1003, [1004](#), [1006](#), [1007](#), T4, T5  
 \@normalsize@check ..... 1006  
 \@noskipsec ..... 651  
 \@noskipsecfalse .....  
     ... r54, r123, r178, O98, X158, X185  
 \@noskipsectrue ..... [O38](#), O95  
 \@notdefinable .....  
     ... f136, f137, f141, f466, [l232](#)  
 \@notprerr ..... [l281](#), r66, r134, r189  
 \@nthm ..... N3, [N4](#)  
 \@nxttabmar ..... 1028, [L11](#), L21,  
     L23, L25, L75, L111, L112, L118, L119  
 \@obsoletefile ..... r649  
 \@oddfoot .....  
     ... S11, S14, S15, X124, X610, X669  
 \@oddhead . S11, S14, X123, X610, X669  
 \@onefilewithoptions .....  
     ... 855, [1059](#), T730, T734, T740,  
     T758, T762, T768, T785, T789,  
     T795, T811, T812, T890, T957, [T1485](#)  
 \@onefilewithoptions@clashchk .....  
     ... T825, [T872](#)  
 \@onelevel@sanitize .....  
     ... 1017, [f697](#), P42, P111  
 \@onfilewithoptions ..... 821

**File Key:** a=ltirchck.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\@onlypreamble ..... 999, 1002, 1009, 1022, 1024, 1026, 1044, 1057, f66, f188, f190, f199, f207, h1450, r196, r205, r242, r651, r677, s23, s24, s61, s62, s66, s125, s145, s189, s190, s204, v17, v115, v117, v123, v139, v167, v182, v203, v208, v250, v512, x419, y28, y36, y42, y79, y83, y88, y93, y98, y108, y126, y127, y128, y134, y138, y142, z17, z19, z44, z46, z107, z116, z136, z353, z354, z362, z402, z440, z442, z444, z457, z472, z519, z521, z563, z602, z618, z695, z789, z798, z854, z857, z860, z880, z893, z947, z982, z993, z1007, z1061, z1081, z1085, z1149, D140, D141, E58, E59, E60, E870, G30, Q12, Q29, R64, R81, T10, T13, T85, T112, T120, T122, T154, T294, T357, T363, T428, T431, T432, T443, T444, T445, T470, T476, T489, T526, T544, T564, T584, T608, T613, T617, T620, T628, T649, T652, T662, T681, T694, T699, T705, T714, T719, T807, T890, T1015, T1024, T1032, T1033, T1051, T1058, T1067, T1074, T1075, T1083, T1088, T1093, T1465, T1466, T1467, T1468, T1470, U140, U141  
 \opargbegintheorem ..... N32, N35  
 \opcol ..... X262, X270, X394, X413, X442, X460, X465  
 \options ..... T563  
 \othm ..... N3, N20  
 \outerparskip J8, J88, J117, J152, J222  
 \outputbox .... 1047, X120, X483, X485, X505, X508, X509, X529, X531, X532, X537, X540, X545, X547, X554, X560, X562, X636, X695, X723, X729, X739, X740, X763, X770, X856, X859, X862, X868, X869, X2237, X2241, X2242, X2256, X2262, X2282, X2288, X2297  
 \outputdblcol ..... X468, X2232, X2234, X2278, X2279  
 \outputpage .. X403, X452, X470, X590, X2266, X2271, X2304, X2312  
 \oval ..... M453, M471  
 \ovbtrue ..... M476, M504, M534  
 \ovdx ..... M431, M487, M489, M495, M497, M516, M518, M526, M528, M543, M551, M553, M589, M592, M599, M601, M693, M694, M695, M696, M712, M713, M714, M717, M733, M753, M754, M755, M756, M772, M773, M775, M789  
 \ovdy ..... M431, M488, M490, M496, M497, M517, M519, M527, M528, M544, M552, M553, M564, M569, M577, M581, M700, M701, M702, M703, M718, M719, M720, M723, M737, M760, M761, M762, M763, M776, M777, M779, M793  
 \ovhlinefalse ..... M477, M505  
 \ovlinetrue ..... M456, M460, M465, M483, M489, M511, M518  
 \ovhorz ..... M494, M495, M525, M526, M550, M551, M584  
 \ovltrue ..... M476, M504, M534  
 \ovri .. K33, M431, M486, M515, M542, M564, M577, M593, M602  
 \ovro ..... M431, M486, M495, M496, M515, M526, M527, M542, M551, M552, M563, M569, M576, M581, M588, M598, M614, M621, M630, M639  
 \ovrtrue ..... M476, M504, M534  
 \ovttrue ..... M476, M504, M534  
 \ovvert ..... M492, M493, M521, M523, M546, M548, M557  
 \ovvlinefalse ..... M477, M505  
 \ovvlinetrue ..... M459, M482, M490, M510, M519  
 \ovxx . M431, M480, M482, M483, M487, M493, M494, M508, M510, M511, M516, M523, M525, M537, M539, M543, M548, M550, M588, M598, M690, M691, M692, M696, M705, M706, M715, M716, M717, M732, M750, M751, M752, M756, M765, M766, M774, M775, M788  
 \ovyy . M431, M481, M482, M483, M488, M495, M509, M510, M511, M517, M526, M538, M539, M544, M551, M561, M574, M697, M698, M699, M703, M705, M721, M722, M723, M736, M757, M758, M759, M763, M765, M778, M779, M792  
 \opfilename ..... T90, T97, T105, T108, T115, T120  
 \opfilepath ..... T91, T138, T147  
 \opfilepath@aux .. T139, T140, T148  
 \pagedp ..... X117, X308, X313, X1091, X1244, X1841, X1851  
 \pageht ..... X116, X309, X313, X315, X316, X317, X321, X1090, X1243, X1824, X1831

**File Key:** a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr1.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisenc.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\@par ..... m3, m5  
 \@parboxrestore .... 1032, K266, K322, K347, K378, K396, K413, P19, P100, P169, P342, P360, P470, P489, P507, X219, X601, X659  
 \@parboxto ..... 1040, K261  
 \@parmoderr .... 1271, P58, P127, P316  
 \@parse@version ..... c80, c101, c102, c114, c115, c162, c163, c164, T195, T201, T202, T212, T1520, T1535, T1572, T1648, T1672  
 \@parse@version@ ..... T189, T190, T195, T207  
 \@parse@version@dash 1045, T213, T215  
 \@partaux ..... r3, r222, r266, r284, r308, r310, r311, r331, r363, r365, r366, r372, r384, r386, r389  
 \@partlist ..... r231, r235, r236, r238, r262, r281, r304, r359  
 \@partswfalse ..... r6  
 \@partswtrue ..... r230, r260, r280  
 \@pass@ptions ..... T380, T382, T402, T403, T405, T418, T419, T421, T428, T429, T430, T901, T983  
 \@pboxswfalse ..... K264, K340  
 \@pboxswtrue ..... K274  
 \@pdef ..... 1002  
 \@openup ..... I186, I187  
 \@percentchar ..... a109, T1178, T1180, T1182, T1184, T1311, T1313, T1315, T1317, T1400, T1402, T1404, T1406, T1454  
 \@pgbk ..... 1026  
 \@picbox 994, M6, M31, M43, M51, M52  
 \@picht ..... M6, M29, M42, M51  
 \@picture ..... M23, M24  
 \@picture@warn ..... M223, M441, M445, M449  
 \@pkgextension ..... 1002  
 \@pkgetension ..... 819, T31, T155, T163, T219, T429, T619, T622, T661, T670, T741, T769, T796, T912, T939, T1069, V483, V489  
 \@plus ..... f26, p456, O16, O206, O229, S54, X2340, X2341, X2342, X2345, X2346, X2350, X2351, X2352, X2356, X2357, X2358  
 \@pnumwidth ..... O218, O241  
 \@popfilename ..... 797, 798, 817, T33, T870, T1005  
 \@pr@videopackage .... T293, T307, T310, T331, T333, T344, T346, T357  
 \@preamble ..... L190, L192, L200, L237, L256, L258, L259, L263, L278, L296, L297, L334  
 \@preamblecmds ..... 785, 1002, f66, r67, r135, r190, T1686, T1687  
 \@preamerr ..... 1260, L199, L274, L355  
 \@process@ptions .... T488, T506, T523, T530, T531, T544, T548, T550  
 \@process@ptions ... T475, T477, T489  
 \@protect@... ..... 1013  
 \@protected@testopt f89, f101, f580, f592  
 \@protecteddef ..... 1008, 1011  
 \@providesfile a101, a102, T367, Z625  
 \@optionlist ..... T152, T222, T474, T878, T884, T973, T979, T1070  
 \@pushfilename ..... 250, 797, 798, T33, T818, T960  
 \@put ..... M452, M497, M528, M553, M621, M639  
 \@qend ..... f136, f671, l236  
 \@qrelax ..... f137, f671  
 \@raw@classoptionslist .... T11, T728  
 \@rc@ifdefinable . f130, f132, f238, s14  
 \@reargdef ..... 1002, f122  
 \@refundefined ..... 1028, r55, r124, r179, G3, H47, H87  
 \@reinserts ..... X327, X330, X516  
 \@remove@eq@value .. T446, T511, T571  
 \@remove@tlig ..... s996, s1004  
 \@remove@tlig@ ..... s996, s997  
 \@remove@tlig@@ ..... s997, s1000  
 \@remove@tlig@@@ ..... s1010, s1029  
 \@removeelement .... k32, T570, T579  
 \@removefromreset ..... t46, t70, t71, t81, t104, t107  
 \@renewfontswitch ..... 1002, 1005  
 \@reqcolroom ..... X1090, X1091, X1094, X1096, X1097, X1102, X1106, X1108, X1136, X1137, X1243, X1244, X1248, X1251, X1252, X1257, X1264, X1266, X1298, X1299, X1410, X1412, X1414, X1417, X1419, X1493, X1496, X1499, X1504, X1506, X2020, X2137, X2142, X2145  
 \@reserved ..... 445  
 \@reset@ptions ..... T834, T871, T899, T964, T1006, T1016  
 \@resetactivechars ..... X575, X598, X656  
 \@resethfps 1009, X1205, X1374, X2087  
 \@resetprotect ..... 1023  
 \@restorepar ..... 1025, m5, p341, p357, p375, p391, J127, J135  
 \@reversemarginfalse ... P388, X101

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntr1.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=lt hyphen.dtx, Z=ltfinal.dtx

\@reversemargintrue ..... P387  
 \@rightmark ..... S16, S51  
 \@rightskip ..... . H363, H382, H400, J75, K295, K316  
 \@rjfieldfalse ..... L34, L77  
 \@rjfieldtrue ..... L125  
 \@rmfamilyhook A429, A451, A463, A490  
 \@roman ..... t137, t143  
 \@rsbox ..... K456, K463, K467  
 \@rtab ..... L71, L86  
 \@rule ..... K428, K433, K437  
 \@sanitize ..... f695, Q7, Q18, Q24, Q35  
 \@savebox ..... K122, K129, K135  
 \@savemarbox ..... P326, P327, P330, P333  
 \@savepicbox ..... K122, K129, K139  
 \@savsf ..... p123, p129, p138, p157, p171, p183, p197, p211  
 \@savsk ..... p123, p128, p139, p158, p172, p184, p198, p212  
 \@scolelt ..... X788, X853  
 \@sdblcolelt ..... X805, X825, X854  
 \@secCntformat ..... 996, 1023, O60, O111  
 \@secondoftwo ..... a91, c84, c182, e36, e40, f212, f445, f539, f589, f640, f647, f667, f714, r451, r496, r513, s131, t190, t195, A512, E71, E838, E889, E905, G21, S17, T159, T191, T203, T236, T254, T1677, Z631  
 \@secpenalty ..... p17, Q36, O50  
 \@sect ..... O54, O55  
 \@seqnrcr ..... I412  
 \@set@curr@file@aux ..... V364  
 \@setckpt ..... 996, 1026, r384, r391, H23, H76  
 \@setfloattypecounts ..... X1074, X1224, X1395, X1476, X1571, X1692, X2034  
 \@setfontsize ..... A637  
 \@setfps ..... P34  
 \@setfpsbit ..... P73, P75, P77, P85, P143, P146, P149, X2078  
 \@setmarks ..... X2248, X2250, X2265  
 \@setminipage ..... . 1020, K352, P21, P177, P185, P376  
 \@setminipage ..... 1033  
 \@setnobreak ..... 1020, P179, P375  
 \@setpar ..... m3, J78  
 \@setprotect ..... 1023  
 \@setref ..... G10  
 \@setspace ..... A637  
 \@settab ..... L71, L93  
 \@settodim ..... u17  
 \@settopoint ..... u22  
 \@setupverbvisiblespace ..... . H473, H484, H496, H513, H528  
 \@sffamilyhook A429, A456, A464, A491  
 \@sharp ..... L196, L235, L265, L280, L281, L301, L303, L305, L333  
 \@shipoutsetup ..... X590  
 \@shortstack ..... M132, M133  
 \@show@ ..... . 97  
 \@show@DeclareRobustCommand ..... . f507, f521, f566, f617, f620  
 \@show@newcommand ..... . 99, f508, f521, f574, f600, f617, f623  
 \@show@newcommand@aux ..... 100, f600, f627  
 \@showcommandlisthook ..... . 97, 143, 146, f504, f506, f514, g1223  
 \@skipping@modulefalse ..... c151, c170, c197  
 \@skipping@moduletrue ..... c150, c174  
 \@sline ..... M167, M179, M184, M269  
 \@slowromancap ..... t144, t145  
 \@spaces ..... l218  
 \@specialoutput ..... X256  
 \@specialpagefalse ..... X97, X607, X665  
 \@specialpagetrue ..... S9  
 \@specialstyle ..... S9, X607, X665  
 \@sp token ..... f680, f690  
 \@sqrt ..... I342  
 \@ssect ..... O53, O112  
 \@stackcr ..... M139, M142  
 \@star@or@long ..... f72, f77, f124, f146, f152, f178, f187, f221  
 \@startcolumn ..... X263, X270, X775  
 \@startdblcolumn ..... . X775, X2270, X2273, X2310, X2316  
 \@startfield ..... . L28, L46, L92, L104, L125, L133  
 \@startline ..... . L20, L57, L62, L68, L83, L154  
 \@startpagehook ..... 999  
 \@startpbox ..... . L193, L236, L266, L332, L384, L386  
 \@startsection ..... O39  
 \@starttoc ..... O149  
 \@stopfield ..... L32, L48, L86, L93, L125, L127, L140, L142, L154  
 \@stopline ..... L30, L56, L85  
 \@stpelt ..... t20, t23  
 \@string@makeletter ..... 104, f701  
 \@strip@args ..... s76  
 \@strip@tex@ext ..... . 338, r226, r236, r238, r243, r273  
 \@strip@tex@ext@aux ..... r243, r274  
 \@svector ..... M244, M259, M269  
 \@sverb ..... . 632, 996, H500, H563, H572, H575  
 \@svsec ..... 996, O57, O60, O66, O78  
 \@svsechd ..... O76, O101, O121

**File Key:** a=ltDIRchk.dtx, b=ltPLAIN.dtx, c=ltVERS.dtx, d=ltLUATEX.dtx, e=ltEXPL.dtx, f=ltDEFNS.dtx, g=ltCMD.dtx, h=ltHOOKS.dtx, i=ltCMDBOOKS.dtx, j=ltALLOC.dtx, k=ltCNTRL.dtx, l=ltERROR.dtx, m=ltPAR.dtx, n=ltPARA.dtx, o=ltMETA.dtx, p=ltSPACE.dtx, q=ltLOGOS.dtx, r=ltFILES.dtx, s=ltOUTENC.dtx, t=ltCOUNTS.dtx, u=ltLENGTH.dtx, v=ltFSSBAS.dtx, w=ltFSSAXES.dtx, x=ltFSSTRC.dtx, y=ltFSSCMP.dtx, z=ltFSSDCL.dtx, A=ltFSSINI.dtx, B=fontdef.dtx, C=preload.dtx, D=ltFNTCMD.dtx, E=ltTEXTCOMP.dtx, F=ltPAGENO.dtx, G=ltXRREF.dtx, H=ltMISCEN.dtx, I=ltMATH.dtx, J=ltLISTS.dtx, K=ltBOXES.dtx, L=ltTAB.dtx, M=ltPICTUR.dtx, N=ltTHM.dtx, O=ltSECT.dtx, P=ltFLOAT.dtx, Q=ltIDXGLO.dtx, R=ltBIBL.dtx, S=ltPAGE.dtx, T=ltCLASS.dtx, U=ltKEYS.dtx, V=ltFILEHOOK.dtx, W=ltSHIPOUT.dtx, X=ltOUTPUT.dtx, Y=ltHYPHEN.dtx, Z=ltFINAL.dtx

\@swaptwoargs . . . . .  
     r530, r532, T844, V153, V167, V187  
 \@sxverbatim . . . . . H408, H485, H492  
 \@tabacckludge . . . . . 1028, 1050, s223,  
     s225, s454, s455, E193, E200, E202  
 \@tabacol . . . . . L178, L258  
 \@tabarray . . . . . L170, L180, L181  
 \@tabclassiv . . . . . L180, L330  
 \@tabclassz . . . . . L179, L282  
 \@tabcr . . . . . L56, L73  
 \@tabfbox . . . . . L16, L80, L82  
 \@tablab . . . . . L72, L126  
 \@tabminus . . . . . L72, L117  
 \@tabplus . . . . . L72, L110  
 \@tabpush . . . . .  
     . . . . . L11, L77, L85, L140, L143, L145  
 \@tabrj . . . . . L72, L124  
 \@tabular . . . . . L174, L177, L178  
 \@tabularcr . . . . . L180, L208  
 \@tempa . . . . . 1020, G47, G48, G59, G60  
 \@tempboxa . . . . . 995,  
     j13, s69, u17, u18, J205, J211, J212,  
     J214, K29, K30, K31, K32, K37,  
     K38, K39, K40, K175, K205, K212,  
     K222, K343, K366, K473, K474,  
     K475, K482, K483, K484, K485,  
     M304, M305, M447, M448, M486,  
     M491, M496, M497, M515, M520,  
     M527, M528, M542, M545, M552,  
     M553, M614, M615, M620, M621,  
     M630, M631, M638, M639, M724,  
     M742, M780, M798, O138, O139,  
     P322, P380, X305, X377, X382,  
     X383, X424, X429, X430, X566,  
     X626, X633, X634, X685, X692,  
     X693, X721, X725, X737, X743,  
     X750, X751, X752, X753, X757, X765  
 \@tempcnta . . . . . j7, z862, z863, z864,  
     z865, z869, L242, L243, L244, L245,  
     M187, M188, M214, M215, M216,  
     M229, M230, M231, M232, M239,  
     M240, M254, M255, M270, M271,  
     M276, M278, M279, M280, M281,  
     M282, M285, M287, M288, M289,  
     M290, M291, M292, M293, M294,  
     M295, M296, M335, M336, M337,  
     M338, M339, M360, M361, M362,  
     M363, M364, M365, M392, M393,  
     M394, M395, M396, M414, M416,  
     M418, M419, M421, M423, M438,  
     M439, M440, M442, M444, M446,  
     M448, M562, M567, M575, M579,  
     M616, M617, M618, M619, M632,  
     M633, M635, M636, M658, M659,  
     M660, M661, M662, M663, M711,  
     M731, M771, M787, P62, P68, P70,  
     P79, P80, P90, P91, P131, P137,  
     P139, P152, P153, P159, P160,  
     T1126, T1128, T1129, T1130,  
     T1257, T1259, T1260, T1261,  
     T1371, T1373, T1374, T1375, X16,  
     X18, X20, X934, X935, X936,  
     X937, X957, X958, X959, X960,  
     X982, X985, X1018, X1021, X1132,  
     X1294, X1574, X1577, X1695,  
     X1698, X1813, X1815, X1818,  
     X1820, X1822, X1844, X2068,  
     X2069, X2073, X2079, X2083,  
     Z213, Z218, Z219, Z220, Z350,  
     Z352, Z353, Z354, Z361, Z363,  
     Z364, Z365, Z371, Z373, Z374,  
     Z375, Z382, Z384, Z385, Z386,  
     Z412, Z414, Z415, Z416, Z438,  
     Z440, Z441, Z442, Z448, Z450,  
     Z451, Z452, Z481, Z486, Z487, Z488  
 \@tempcntb . . . . .  
     j7, z863, z867, z869, M279, M280,  
     M281, M283, M284, M285, M562,  
     M563, M567, M568, M575, M576,  
     M579, M580, P88, P89, P90, P157,  
     P158, P159, X17, X20, X21, X2079,  
     X2080, X2081, Z214, Z218, Z482, Z486  
 \@tempdima . . . . . 996, j10, v219,  
     v224, K43, K44, K204, K205, K210,  
     K211, K212, K214, K265, K266,  
     K341, K345, K440, K443, K444,  
     K471, K473, K479, K482, L35, L36,  
     L37, L88, L89, L90, L91, L218,  
     L219, M210, M211, M213, M214,  
     M215, M216, M217, M218, M437,  
     M438, M439, M448, M487, M488,  
     M492, M493, M516, M517, M521,  
     M523, M543, M544, M546, M548,  
     M617, M619, M634, M636, M637,  
     M657, M658, M659, O211, O212,  
     O234, O235, O248, P196, P198,  
     P218, P220, P258, P259, P260,  
     X229, X230, X231, X487, X489,  
     X535, X537, X538, X543, X548,  
     X552, X557, X561, X917, X920,  
     X940, X950, X962, X972, X1637,  
     X1638, X1641, X1642, X1762,  
     X1763, X1767, X1768, X1823,  
     X1824, X1825, X1826, X1829,  
     X1832, X1835, X1837, X2186,  
     X2187, X2189, X2190, I173, I176, I182  
 \@tempdimb . . . . . 996, 1040,  
     j10, v220, v225, v624, v628, x179,

**File Key:** a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrn.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

x180, x437, x460, x461, x470, x471,  
 x475, x493, x496, x499, x501, K268,  
 K269, K441, K444, K472, K474,  
 K480, K483, M211, M212, M357,  
 M359, M362, M365, M482, M484,  
 M485, M510, M513, M514, M539,  
 M540, M541, M612, M613, M622,  
 M628, M629, M640, M648, M649,  
 M654, X940, X941, X942, X943,  
 X950, X962, X963, X964, X965, X972  
 \@tempdimc ..... j10, x454, x455,  
 x457, x458, x460, x461, K53, K54,  
 K64, K65, K442, K443, K444, M30,  
 M31, M32, M33, M34, M35, M60,  
 M61, M63, M64, M332, M333, M334  
 \@tempdimx ..... 996  
 \@tempskipa ..... j14, p44,  
 p47, p48, p285, p292, p294, p297,  
 x181, x182, J116, J117, J118, J150,  
 J152, J153, J154, J222, J223, J224,  
 O42, O44, O45, O50, O62, O63,  
 O88, O89, O91, O103, O104, O113,  
 O114, X1872, X1873, X1875, X1883  
 \@tempskipb .....  
 . 322, j14, p220, p222, p224, p227,  
 p229, p243, p261, p283, p285, p286,  
 p290, p292, p294, p295, p318, p321  
 \@tempswa ..... 992  
 \@tempswafalse ..... a81, b262,  
 r302, r357, s1537, v94, v673, z404,  
 z459, z523, z604, z1124, z1130, H25,  
 H78, H423, H444, R17, R29, T1103,  
 T1119, T1234, T1250, V287, V310,  
 V331, X988, X1024, X1580, X1701  
 \@tempswatrue ..... a82,  
 b268, r300, r305, r355, r360, s1540,  
 s1541, v97, v674, v675, v678, v681,  
 z407, z462, z526, z607, z1087, H121,  
 H428, H449, R17, R29, T1100,  
 T1231, V287, V310, V331, X1582,  
 X1605, X1703, X1728, X2147, X2164  
 \@temptokena .....  
 619, j16, H133, H137, H146, H159,  
 H160, S26, S27, S34, S35, S48, S49  
 \@test... ..... 1031  
 \@test@opt ..... 1027  
 \@testdef ..... H24, H77, H119  
 \@testfalse ..... X12, X14, X15  
 \@testfp ..... 1019, X882, X902,  
 X938, X961, X2071, X2200, X2217  
 \@testopt ..... 1027, f33, f79,  
 f99, f103, f148, p7, p8, p9, p10, I383  
 \@testpach ..... L270, L348  
 \@testpatch ..... L348

\@testtrue ..... X13, X21,  
 X356, X885, X904, X944, X966, X2075  
 \@testwidth ..... X345,  
 X883, X939, X1112, X1426, X1631  
 \@text@composite 1021, s76, s1070, s1075  
 \@text@composite@x ..... s76  
 \@textbottom .....  
 .. S54, S56, X511, X549, X563, X572  
 \@textfloatsheight .....  
 .. X476, X1087, X1089,  
 X1139, X1140, X1145, X1240,  
 X1242, X1302, X1304, X1310, X2020  
 \@textmin ..... P285, P286,  
 P299, P300, X112, X1089, X1093,  
 X1096, X1097, X1242, X1247,  
 X1251, X1252, X1414, X1499,  
 X1598, X1600, X1616, X1720,  
 X1722, X1740, X2128, X2130, X2132  
 \@textsubscript ..... P424, P432  
 \@textsuperscript .....  
 .. 1024, P401, P403, P404  
 \@texttop . S54, S56, X507, X530, X572  
 \@tf@r ..... 1005, k25, k26  
 \@tfor ..... k25,  
 r497, r514, r658, D88, K57, K76,  
 L268, M478, M506, M535, P63, P132  
 \@tf@loop ..... k27, k28, k30  
 \@thanks ..... O11, O34  
 \@thefnmark K381, K398, K415, P400,  
 P401, P453, P458, P473, P491,  
 P509, P521, P526, P537, P542, P553  
 \@thefoot ..... X124,  
 X610, X613, X640, X669, X672, X699  
 \@thehead ..... X123,  
 X610, X612, X630, X669, X671, X689  
 \@themargin ..... X74,  
 X611, X613, X625, X670, X672, X684  
 \@themark . S25, S26, S33, S34, S49, S52  
 \@thirdofthree ..... f216, s197  
 \@thm ..... N12, N18, N24, N26  
 \@thmcounter ..... N11, N17, N33  
 \@thmcountersep ..... N10, N33  
 \@title ..... O7, O31  
 \@tocrmarg ..... O207, O230  
 \@toodeep ..... l253, J36, J232, J243  
 \@toplisp ..... X64, X384, X385,  
 X431, X432, X716, X722, X732,  
 X733, X1025, X1037, X1954, X1982  
 \@topnewpage ..... 1008, X199  
 \@topnum P271, X105, X1022, X1023,  
 X1037, X1041, X1458, X1463,  
 X1551, X1558, X1945, X1973, X2014  
 \@toproom ..... P273,  
 X106, X1025, X1037, X1946, X1974

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\@topsep ..... [J1](#), [J71](#), [J73](#), [J171](#)  
 \@topsepadd .... [J1](#), [J59](#), [J61](#), [J71](#), [J124](#)  
 \@totalleftmargin .....  
     ..... [651](#), [H419](#), [H441](#), [J9](#),  
[J53](#), [J54](#), [K294](#), [K315](#), [L35](#), [L76](#), [L81](#)  
 \@trivlist ..... [J48](#), [J57](#), [J92](#)  
 \@tryfcolumn ..... [X778](#),  
[X798](#), [X816](#), [X832](#), [X2201](#), [X2218](#)  
 \@trylist [X841](#), [X844](#), [X877](#), [X897](#), [X919](#)  
 \@ttfamilyhook [A429](#), [A461](#), [A465](#), [A492](#)  
 \@twoclasseserror ..... [T610](#), [T1089](#)  
 \@twocolumnfalse ..... [X99](#), [X147](#)  
 \@twocolumntrue ..... [X206](#)  
 \@twoheadclasserror .....  
     ..... [T869](#), [T1004](#), [T1084](#)  
 \@twosidefalse ..... [X100](#)  
 \@typein ..... [1027](#), [f32](#), [f33](#), [f40](#), [f48](#)  
 \@typeset@protect . [89](#), [1019](#), [1025](#),  
[f102](#), [f243](#), [f250](#), [f252](#), [s26](#), [s32](#), [s210](#),  
[s218](#), [A638](#), [H193](#), [H238](#), [H256](#), [Z344](#)  
 \@uclclist .....  
     ..... [1034](#), [1037](#), [s1506](#), [s1507](#), [s1560](#), [Z551](#)  
 \@undefined .....  
     ... [M57](#), [M71](#), [M82](#), [M91](#), [M162](#),  
[M174](#), [M237](#), [M252](#), [M313](#), [M371](#)  
 \@undefined ... [868](#), [1046](#), [a71](#), [a72](#),  
[a111](#), [a112](#), [a113](#), [a134](#), [a142](#), [a150](#),  
[a157](#), [a208](#), [a212](#), [a238](#), [a245](#), [a329](#),  
[a330](#), [b65](#), [b81](#), [b105](#), [b106](#), [b121](#),  
[b122](#), [b127](#), [b136](#), [b149](#), [b184](#), [b189](#),  
[b222](#), [b223](#), [b246](#), [b256](#), [b291](#), [b349](#),  
[b359](#), [b361](#), [b442](#), [b443](#), [b453](#), [b454](#),  
[b549](#), [b617](#), [b657](#), [b672](#), [c50](#), [c59](#),  
[d2](#), [d15](#), [d16](#), [d17](#), [d29](#), [d30](#), [d74](#),  
[d84](#), [d173](#), [d189](#), [d197](#), [d205](#), [d237](#),  
[d238](#), [d239](#), [d240](#), [d241](#), [d242](#), [d243](#),  
[d244](#), [d245](#), [d246](#), [d247](#), [d248](#), [d249](#),  
[d250](#), [d251](#), [d252](#), [d253](#), [d254](#), [d255](#),  
[d261](#), [e4](#), [e13](#), [e14](#), [e15](#), [e16](#), [e147](#),  
[e148](#), [e149](#), [e150](#), [e151](#), [e152](#), [e169](#),  
[e170](#), [e171](#), [e172](#), [e192](#), [e193](#), [f34](#),  
[f223](#), [f307](#), [f308](#), [f339](#), [f367](#), [f371](#),  
[f372](#), [f389](#), [f403](#), [f450](#), [f451](#), [f452](#),  
[f453](#), [f454](#), [f455](#), [f456](#), [f490](#), [f491](#),  
[f492](#), [f493](#), [f494](#), [f495](#), [f513](#), [f514](#),  
[f618](#), [f619](#), [f620](#), [f621](#), [f622](#), [f623](#),  
[f625](#), [f626](#), [f627](#), [f642](#), [f649](#), [f723](#),  
[f724](#), [f725](#), [f734](#), [l28](#), [n128](#), [n129](#),  
[n130](#), [n131](#), [n132](#), [o22](#), [p89](#), [p470](#),  
[p476](#), [r61](#), [r62](#), [r129](#), [r130](#), [r184](#),  
[r185](#), [r273](#), [r274](#), [r422](#), [r423](#), [r424](#),  
[r429](#), [r450](#), [r471](#), [r486](#), [r541](#), [s195](#),  
[s197](#), [s333](#), [s335](#), [s337](#), [s339](#), [s341](#),  
[s343](#), [s345](#), [s347](#), [s349](#), [s351](#), [s370](#),  
     ..... [s372](#), [s374](#), [s458](#), [s700](#), [s703](#), [t204](#),  
[v434](#), [v474](#), [v536](#), [v569](#), [v633](#), [v640](#),  
[w388](#), [w411](#), [w420](#), [w509](#), [w510](#),  
[w511](#), [w512](#), [w513](#), [w514](#), [w515](#),  
[w516](#), [w517](#), [w518](#), [w529](#), [w534](#),  
[w539](#), [w603](#), [w604](#), [w605](#), [w606](#),  
[w607](#), [w608](#), [w609](#), [w634](#), [w714](#),  
[w716](#), [w717](#), [w719](#), [w720](#), [w721](#),  
[w723](#), [x552](#), [x553](#), [y4](#), [y5](#), [y6](#), [y7](#),  
[y8](#), [y9](#), [y10](#), [y11](#), [y12](#), [y13](#), [y14](#), [y15](#),  
[y16](#), [y17](#), [y18](#), [y19](#), [y20](#), [z141](#), [z720](#),  
[z827](#), [A35](#), [A58](#), [A71](#), [A91](#), [A104](#),  
[A105](#), [A106](#), [A107](#), [A108](#), [A109](#),  
[A110](#), [A111](#), [A112](#), [A113](#), [A114](#),  
[A116](#), [A117](#), [A118](#), [A231](#), [A232](#),  
[A233](#), [A234](#), [A235](#), [A236](#), [A237](#),  
[A238](#), [A239](#), [A240](#), [A242](#), [A243](#),  
[A244](#), [A304](#), [A305](#), [A472](#), [A490](#),  
[A491](#), [A492](#), [A534](#), [A535](#), [A536](#),  
[A537](#), [A538](#), [A578](#), [A579](#), [A580](#),  
[A581](#), [A582](#), [A583](#), [A594](#), [A716](#),  
[B15](#), [B50](#), [B65](#), [D37](#), [D38](#), [D39](#),  
[D122](#), [E159](#), [E367](#), [E617](#), [E620](#),  
[E621](#), [E651](#), [E652](#), [E653](#), [E656](#),  
[E657](#), [E658](#), [E659](#), [E660](#), [E661](#),  
[E666](#), [E667](#), [E668](#), [E669](#), [E674](#),  
[E675](#), [E676](#), [E677](#), [E680](#), [E681](#),  
[E682](#), [E684](#), [E685](#), [E690](#), [E691](#),  
[E692](#), [E693](#), [E694](#), [E695](#), [E696](#),  
[E697](#), [E698](#), [E699](#), [E700](#), [E701](#),  
[E702](#), [E703](#), [E704](#), [E705](#), [E707](#),  
[E708](#), [E710](#), [E711](#), [E712](#), [E713](#),  
[E714](#), [E715](#), [E716](#), [E717](#), [E718](#),  
[E719](#), [E721](#), [E722](#), [E723](#), [E724](#),  
[E725](#), [E726](#), [E727](#), [E728](#), [E729](#),  
[E731](#), [E732](#), [E733](#), [E734](#), [E735](#),  
[E736](#), [E737](#), [E738](#), [E739](#), [E740](#),  
[E741](#), [E742](#), [E743](#), [E744](#), [E745](#),  
[E746](#), [E747](#), [E748](#), [E749](#), [E750](#),  
[E751](#), [E752](#), [E753](#), [E754](#), [E755](#),  
[E756](#), [E757](#), [E762](#), [E763](#), [E765](#),  
[E766](#), [E767](#), [E768](#), [E769](#), [E770](#),  
[E771](#), [E772](#), [E774](#), [E775](#), [E777](#),  
[E778](#), [E780](#), [E781](#), [E782](#), [E783](#),  
[E785](#), [E786](#), [E787](#), [E789](#), [E791](#),  
[E792](#), [E793](#), [E794](#), [E795](#), [E796](#),  
[E797](#), [E799](#), [E800](#), [E801](#), [E802](#),  
[E803](#), [E804](#), [E805](#), [E806](#), [E807](#),  
[G53](#), [G69](#), [G70](#), [H101](#), [H153](#), [H154](#),  
[H155](#), [H156](#), [H291](#), [H292](#), [H312](#),  
[H313](#), [H314](#), [H315](#), [H467](#), [H494](#),  
[H495](#), [H496](#), [H497](#), [H530](#), [K21](#),  
[K130](#), [K194](#), [K250](#), [K434](#), [K464](#),  
[M18](#), [M467](#), [M468](#), [O197](#), [O245](#)

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=lxref.dtx, H=lmiscen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=lpictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=lyphen.dtx, Z=ltfinal.dtx

P5, P429, P448, P562, R30, R53, T4, T25, T146, T147, T148, T207, T353, T461, T511, T891, T929, T931, T937, T994, T1009, T1010, T1025, T1139, T1217, T1220, T1270, T1349, T1352, T1362, T1363, T1364, T1365, T1366, T1367, T1368, T1439, T1442, T1458, T1478, T1485, V18, V19, V20, V21, V147, V189, V190, V197, V198, V199, V321, V342, V356, V360, V361, V489, V495, V496, V497, W427, W428, W429, W430, W431, W433, W434, W435, W442, W443, W445, W447, W448, W449, W452, W475, X36, X368, X369, X1924, X1925, X1926, X1927, Z10, Z18, Z25, Z40, Z59, Z68, Z75, Z84, Z94, Z146, Z147, Z256, Z269, Z282, Z302, Z327, Z337, Z338, Z339, Z368, Z370, Z409, Z410, Z429, Z430, Z431, Z432, Z433, Z434, Z435, Z436, Z437, Z454, Z470, Z471, Z472, Z520, Z521, Z579, Z614, Z615, Z616, Z617, Z618, I223, I233, I234, I270, I273, I316, I329  
 \undefinedfonterror ..... 1009  
 \unexpandable@protect ..... 1018, f220, f255, f261, f266, r210, L264  
 \unknownoptionerror ..... T1020, T1059, T1072  
 \unknownverson ..... 998  
 \unprocessedoptions ..... 819, 820, 1032, T560, T669, T891, T930, T931, T991, T995, T1074, U49  
 \unused ..... 1056, b307, f10, f17, l15, l32, l59, T1463  
 \unusedoptionlist ..... 844, 997, r18, r20, r87, r89, r144, r146, T12, T453, T454, T464, T465, T572, T580, U86, U98  
 \upline ..... M297, M298, M304  
 \upordown ..... M195, M196, M204, M225, M273  
 \upvector ..... M268, M304  
 \use@option ..... T484, T500, T518, T534, T536, T553, T555, T565  
 \use@text@encoding ..... s150, E16, E1141  
 \vbsphack ..... p219  
 \verb ..... H563, H572, H575  
 \verbatim ..... H413, H459, H483, H492  
 \verbvisiblebox ..... H478, H479, H482, H497  
 \vereq ..... B457, B458  
 \viiipt ..... v716  
 \viipt ..... v715  
 \vipt ..... v714  
 \vline ..... M166, M178, M297  
 \vobeyspaces ..... H406, H459, H485, H513, H528, H575  
 \vppt ..... v713  
 \vspace ..... p330  
 \vspace@calcify ..... p79, p101, p241, p337, p342, p352, p360, H342, L62, L224, M148, I391  
 \vspacer ..... p330  
 \vtryfc ..... X847, X855  
 \vvector ..... M243, M258, M268  
 \warning ..... l215  
 \wckptelt ..... r385, r388  
 \whiledim ..... k7, M123, M203  
 \whileoop ..... 1026, k3  
 \whilenum ..... k3, L244, M104, M116, M336, M338, M361, M364, M393, M395, M416, M421, M731, M787  
 \whilesw ..... k10, X264, X394, X403, X441, X451, X2271, X2311  
 \whileswnoop ..... 1026, k10  
 \wholewidth ..... K160, K162, K163, K165, K167, K168, K169, K170, M2, M126, M129, M131, M299, M302, M350, M359, M406, M413, M565, M578, M590, M600, M679, M680, M728, M784  
 \width ..... 1022, b472, f26, p452, s289, s294, x192, B618, K165, K167, K218, K225, K444, K488, L188, L219, L347, L366, M227, M299, M302, M325, M333, M350, M359, M384, M392, M406, M413, M565, M578, M728, M784, P395, X1851, X2260, X2294  
 \wrglossary ..... 1018, Q25, Q30  
 \windex ..... 1018, Q8, Q13  
 \writeckpt ..... r329, r370, r382  
 \writefile ..... 619, 752, r32, r101, r159, H140, O183  
 \writesetup ..... X590  
 \wrong@font@char ..... 1029, s161, v537, v571, v584  
 \wtryfc ..... X857, X867  
 \x@protect ..... f105, f242  
 \x@sf ..... P531, P533  
 \xDeclareMathDelimiter ..... z892, z948  
 \xaddvskip ..... p219, p244, p262  
 \xarg ..... M163, M166, M175, M178, M185, M189, M190, M226,

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=lt hyphen.dtx, Z=ltfinal.dtx

M228, M238, M239, M243, M253,  
 M254, M258, M266, M274, M664  
 \@xargarraycr ..... L205, L214, L218  
 \@xargdef ..... f80  
 \@xarraycr ..... L202, L203  
 \@xbitor ..... X15, X17  
 \@xcentercr ..... H325, H332, H336  
 \@xdblarg ..... f693  
 \@xdblfloat ..... 1017, P264  
 \@xdim ..... M84, M93,  
     M105, M107, M117, M119, M668,  
     M732, M733, M734, M735, M741,  
     M788, M789, M790, M791, M797  
 \@xeqncr ..... I375  
 \@xexnoop ..... L238, L248  
 \@xexpast ..... L239, L240  
 \@xfloat .. 1018, P28, P29, P34, P266  
 \@xfootnote ..... 995, P452, P455  
 \@xfootnotemark ..... 995, P519, P523  
 \@xfootnotenext ..... P536, P539  
 \@xfootnotetext ..... 995  
 \@xhline ..... L360, L361  
 \@xifnch ..... f681, f691  
 \@xiipt ..... v720, B164, B166, B167  
 \@xipt ..... v719, B163  
 \@xivpt ..... v721, B165, B167  
 \@xmpar ..... P324, P325  
 \@xnewline ..... p60, p61, p72, p73, p93  
 \@xnext ..... X10, X11  
 \@xnthm ..... N5, N6  
 \@xobeysp .....  
     326, p420, H407, H408, H475, H479  
 \@xprocessoptions .....  
     T475, T492, T494, T510, T512, T526  
 \@xpt ..... v718, B162, B165, B166  
 \@xsect ..... O86, O87, O123  
 \@xtabcr ..... L56, L57  
 \@xtabularcr ..... L209, L210  
 \@xthm ..... N28, N29  
 \@xtryfc ..... X844, X872  
 \@xtypein ..... f33, f35, f42  
 \@xverbatim ..... H408, H459  
 \@xvipt ..... v722, B166, B168  
 \@xxDeclareMathDelimiter . z877, z881  
 \@xxpt ..... v723, B167, B168  
 \@xxvpt ..... v724, B168  
 \@xxxii ..... 1029, j2, s425,  
     s427, P89, P158, X879, X880, X899,  
     X900, X935, X936, X958, X959, X2037  
 \@xympar ..... P328, P332, P378  
 \@yarg .....  
     M163, M167, M175, M179, M185,  
     M186, M195, M238, M244, M253,  
     M259, M268, M270, M297, M664  
 \@yargarraycr ..... L206, L216, L220  
 \@yargd@f ..... f107  
 \@yargdef .. 1005, 1031, f84, f94, f107, f123  
 \@ydim ..... M85, M94,  
     M105, M108, M117, M119, M668,  
     M736, M737, M738, M739, M740,  
     M792, M793, M794, M795, M796  
 \@yeqnacr ..... I375  
 \@ympar ..... P324, P329  
 \@ynthm ..... N5, N14  
 \@ythm ..... N28, N29  
 \@ytryfc ..... X890, X909, X913  
 \@yyarg ..... M185,  
     M186, M187, M190, M274, M664  
 \@ztryfc ..... X918, X929  
 \@zend ..... 202  
 \accent@spacefactor .. 1037, s70, s73, s74  
 \active@math@prime .....  
     1037, X588, I240, I241  
 \add@accent ..... 1030, s65, s67  
 \add@percent@to@temptokena .....  
     620, H127, H143, H145, H154  
 \add@unicode@accent ..... s1039, s1053  
 \addto@hook .....  
     993, 1004, v152, v154, v712, z386,  
     z482, z486, z503, z627, z633, z641,  
     z657, z660, z663, z1096, z1103, z1106  
 \AddToHook ..... 254  
 \AddToHookNext ..... 254  
 \alloc@ ..... 1052,  
     b90, b91, b92, b93, b94, b95, b96,  
     b97, b98, b99, b226, d22, d26, d38, v14  
 \alpha@elt .....  
     z45, z390, z577, z679, z1095, z1096  
 \alpha@list .....  
     z41, z43, z399, z565, z577,  
     z622, z677, z678, z1091, z1097, z1098  
 \apptocmd ..... 254  
 \atveryend@DEPRECATED .. V577, V579  
 \best@size ..... x438, x462, x468, x474  
 \bf@def@ult ..... A397  
 \bfdef@ult ..... 447,  
     524, 533, 1050, A200, A271, A272,  
     A273, A314, A315, A316, A401, A442  
 \bfdefault@previous .. 524, 1053,  
     A267, A270, A310, A313, B106, B116  
 \bfseries@.. ..... 519  
 \bfseries@previous ..... 1053  
 \bfseries@rm ..... 519,  
     522, 533, A105, A128, A207, A210,  
     A231, A258, A271, A314, A319, A355  
 \bfseries@rm@kernel .....  
     522, A108, A131, A207, A234

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfnctcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=lt hyphen.dtx, Z=ltfinal.dtx

\bfseries@sf . . . . .	X44
516, 522, A106, A128, A211, A214, A232, A259, A272, A315, A320, A356	
\bfseries@sf@kernel . . . . .	X44
A109, A132, A211, A235	
\bfseries@tt . . . . .	X44
... 522, A107, A128, A215, A218, A233, A260, A273, A316, A321, A357	
\bfseries@tt@kernel . . . . .	X44
A110, A133, A215, A236	
\bm@b . . . . .	K37
\bm@c . . . . .	K37
\bm@l . . . . .	K37
\bm@r . . . . .	K37
\bm@s . . . . .	K37
\bm@t . . . . .	K37
\bx@ . . . . .	955
\bx@A . . . . .	X30, X57
\bx@AA . . . . .	X40
\bx@B . . . . .	X30, X57
\bx@BB . . . . .	X40
\bx@C . . . . .	X30, X57
\bx@CC . . . . .	X40
\bx@D . . . . .	X30, X57
\bx@DD . . . . .	X40
\bx@E . . . . .	X30, X57
\bx@EE . . . . .	X40
\bx@F . . . . .	X31, X58
\bx@FF . . . . .	X41
\bx@G . . . . .	X31, X58
\bx@GG . . . . .	X41
\bx@H . . . . .	X31, X58
\bx@HH . . . . .	X41
\bx@I . . . . .	X31, X58
\bx@II . . . . .	X41
\bx@J . . . . .	X31, X58
\bx@JJ . . . . .	X41
\bx@K . . . . .	X32, X59
\bx@KK . . . . .	X42
\bx@L . . . . .	X32, X59
\bx@LL . . . . .	X42
\bx@M . . . . .	X32, X59
\bx@MM . . . . .	X42
\bx@N . . . . .	X32, X59
\bx@NN . . . . .	X42
\bx@O . . . . .	X33, X60
\bx@OO . . . . .	X43
\bx@P . . . . .	X33, X60
\bx@PP . . . . .	X43
\bx@Q . . . . .	X33, X60
\bx@QQ . . . . .	X43
\bx@R . . . . .	X33, X60
\bx@RR . . . . .	X43
\bx@S . . . . .	X38
\bx@SS . . . . .	X44
\bx@T . . . . .	X38
\bx@TT . . . . .	X44
\bx@U . . . . .	X38
\bx@UU . . . . .	X44
\bx@V . . . . .	X38
\bx@VV . . . . .	X44
\bx@W . . . . .	X39
\bx@WW . . . . .	X45
\bx@X . . . . .	X39
\bx@XX . . . . .	X45
\bx@Y . . . . .	X39
\bx@YY . . . . .	X45
\bx@Z . . . . .	X39
\bx@ZZ . . . . .	X25, X45, X55
\c@bottomnumber . . . . .	P269, P274, X2326
\c@dbltopnumber . . . . .	P268, P283, P297, X2333
\c@enumi . . . . .	J227
\c@enumii . . . . .	J227
\c@enumiv . . . . .	995, J227
\c@equation . . . . .	I336, I370, I513
\c@errorcontextlines . . . . .	999, l212
\c@footnote . . . . .	O12, P397, P525
\c@localalphabets . . . . .	z137
\c@localmathalphabets . . . . .	490, z141, z142, z150, z183
\c@mpfootnote . . . . .	K348, P399
\c@ncel . . . . .	B461, B462
\c@page . . . . .	F3, F6, F7, X138, X1818
\c@secnumdepth . . . . .	O56, O71, O81, O140
\c@tocdepth . . . . .	O140, O205, O228
\c@topnumber . . . . .	P267, P271, X2322
\c@totalnumber . . . . .	P270, P276, X2329
\c@totalpages . . . . .	W346, W430
\calculate@math@sizes . . . . .	v620, x219
\catcodetable@atletter . . . . .	d93, d246
\catcodetable@initex . . . . .	d93, d243
\catcodetable@latex . . . . .	d93, d245
\catcodetable@string . . . . .	d93, d244
\cdp@elt . . . . .	v96, v116, v127, v128, v149, v152, v154, z311, z406, z461, z525, z606, Z458, Z459
\cdp@list . . . . .	v98, v114, v128, v156, v157, z329, z408, z463, z527, z608, Z459
\cf@encoding . . . . .	s34, s41, s44, s51, s154, v256, v266, v276, v326
\ch@ck . . . . .	b206, b207, b208, b209, b236, b248, b249, b250, b251, b279, b281, b293, b294, b295, b296, b302, T1144, T1163, T1275, T1296, T1388
\char@if@alph . . . . .	f701
\chardef@text@cmd . . . . .	s3
\check@command . . . . .	f187, f189

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=lptra.dtx, o=ltmeta.dtx, p=ltspac.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

```

\check@icl D9, D44, D49, D55, D63, D70, D72
\check@icr 1032, 1034, D9, D44, D50, D56, D64, D73, D78
\check@mathfonts 491, q5, s302, s328, s360, s1254, v348, v350, x250, z188, E672
\check@nocorr 1015
\check@nocorr@ 1034, D46
\check@range x379, x380
\check@single x378, x400
\cl@ckpt r385, t35
\cl@page F4
\col@number 1008, X95, X148, X208, X220
\color@begingroup 1005, 1017, v643, v703, K29, K89, K176, K344, K382, K399, K416, L47, L51, P475, P493, P511, X491, I104, I134
\color@endbox .. K89, P253, P348, P366, X224, X631, X641, X690, X700
\color@endgroup 1005, v648, v709, K29, K89, K134, K155, K178, K364, K386, K403, K419, L49, P479, P497, P514, X495, I104, I134
\color@hbox 1018, 1019, K89, X628, X638, X687, X697
\color@setgroup 1005, 1010, K89, K134, K153
\color@vbox 1020, K89, P96, P165, P339, P357, P381, X215
\conditionally@traceoff 289, g217, g229, l289
\conditionally@traceon ... g253, l289
\contionally@traceon 289
\copy@kernel@robust@command f541, f625
\count@ a69, a182, a183, a184, a189, b41, b191, b192, b197, b199, b205, b206, b207, b208, b209, b210, b472, b473, c14, c15, c16, c17, c18, c20, f169, f173, f275, f300, v677, v683, v685, x22, x302, x304, x326, x327, z383, z385, z389, z708, z709, z710, z756, z757, z758, z817, z818, z819, z865, z866, z867, z905, z906, z907, z913, z914, z915, z959, z960, z961, z967, z968, z969, z1028, z1029, z1030, z1036, z1037, z1038, D115, D118, M730, M731, M732, M735, M736, M739, M743, M786, M787, M788, M791, M792, M795, M799, T1187, T1189, T1190, T1191, T1320, T1322, T1323, T1324, T1409, T1411, T1412, T1413, Z225, Z226, Z233, Z235, Z525, Z526, Z533, Z535
\counterwithin@s t113, t114, t132
\counterwithin@x t113, t116, t133
\counterwithout@s t102, t103, t129
\counterwithout@x t102, t105, t130
\curr@fontshape .. 993, s180, v88, v363, v371, v375, v377, v519, v525, v528, v537, v544, v546, v554, v560, v563, v571, v578, v580, w454, x92, x100, x142, x169, x477, x497, x529, x545, x560, z333, z338, A556, A565
\curr@math@size v352, x256, x262, x267, x284
\declare@command@copy 1052
\declare@commandcopy 96, f464, f468, f473, f476, f493
\declare@commandcopy@let 96, f481, f485, f495, f563, f564
\declare@file@substitution 852, V247, V564, W506
\declare@robustcommand f221
\DeclareEncodingSubset@aux E41, E43, E59
\DeclareFontEncoding@ 1046, v122, v124, v139, Z369, Z389, Z455
\DeclareFontEncoding@saved Z369, Z389, Z471
\DeclareFontShape@ v21, v22
\DeclareRobustCommand 255, 262
\DeclareSymbolFontAlphabet@ z1083, z1086
\DeclareUnicodeAccent@ s1046, s1048, s1052
\def 255, 257, 269, 270
\default@ds 997, T442, T471, T535, T554, T1018, T1020
\default@family .. 992, v129, v161, v487, v501, v504, v529, v564, Z460
\default@M 997, v136, v176, v179, v183, Z467
\default@mextra y10, y89
\default@series v129, v162, v488, v502, v505, v526, v561, Z460
\default@shape v130, v163, v489, v503, v506, v524, v559, Z461
\default@T .. 997, v170, v173, v183, v272
\define@mathalphabet .. 992, y18, y131
\define@mathgroup 992, y19, y135
\define@newfont v355, v364
\delayed@f@adjustment 416, 443, 460, 538, v290, w395, w396, w397, w399,

```

**File Key:** a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrnl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

w400, w411, w616, w617, w619,  
w620, x122, x126, x134, x138, A662  
\delayed@merge@font@series .....  
.. w396, w460, w475, w514, x133, x136  
\delayed@merge@font@shape .....  
.. w617, w666, w721, x132, x135  
\development@branch@name .....  
. c11, c36, c50, c51, c52, c59, c60, c61  
\dimen@ .....  
1031, b41, b469, b470, b506, b507,  
b509, b511, l28, l29, p349, p354,  
p383, p388, s429, s430, s432, s433,  
s796, s797, v214, v216, v222, v235,  
v238, v242, v623, v624, v625, v629,  
x451, x452, x453, x454, x458, E102,  
E104, E931, E933, K481, K484,  
L176, L177, X508, X510, X531,  
X533, I72, I73, I186, I187, I188, I189  
\dimen@i ..... b41  
\dimen@ii ..... b41, v218, v223  
\disable@package@load .....  
.. 852, 867, V478, W464  
\displ@y ..... I191, I195, I196  
\do@add@percent@to@temptokena ...  
..... H131, H137, H155  
\do@emfont@update .. A557, A561, A581  
\do@noligs ..... H577, H582  
\do@subst@correction .. v84, x482, x537  
\document@default@language ...  
.. r51, r52, r120, r121, X597, Z272  
\document@select@group ...  
.. 490, z139, z145, z346  
\dont@add@percent@to@temptokena ...  
..... H130, H132, H156  
\dorestore@version ..... z114, z119  
\ds@ ..... 997, T473, T1022  
\dt@pfalse ..... I192  
\dt@ptrue ..... I191  
\e@alloc .....  
1052, b51, b52, b53, b55, b56, b63,  
b64, b66, b68, b79, b82, b84, b138,  
b230, d15, d49, d79, d89, d178,  
d182, d186, d194, d202, d210, Z12, Z33  
\e@alloc@attribute@count .....  
.. d66, d74, d75, d76, d80, d237  
\e@alloc@bytecode@count .....  
.. d70, d197, d198, d199, d203, d253  
\e@alloc@ccodetable@count .....  
.. d67, d84, d85, d86, d90, d241  
\e@alloc@chardef .. b60, b102, b210,  
b211, d48, d178, d194, d202, d210, Z12  
\e@alloc@intercharclass@top ... Z21  
\e@alloc@luachunk@count .....  
.. d71, d205, d206, d207, d211, d255  
\e@alloc@luafunction@count .....  
.. d68, d173, d174,  
d175, d179, d183, d187, d247, d249  
\e@alloc@top ..... b55, b63,  
b102, b188, b259, d47, d80, d179,  
d183, d187, d195, d203, d211, Z12  
\e@alloc@whatsit@count .....  
.. d69, d189, d190, d191, d195, d251  
\e@ch@ck ..... b142, b152, d51, d55  
\e@insert@top .. b257, b259, b276, b291  
\e@mathgroup@top ..... 490, 1042,  
b79, b124, z56, z149, z150, z224, z254  
\em@currfont ..... 535, A556, A567  
\em@force 535, A565, A570, A573, A583  
\emfontdeclare@clist .. 534, A546,  
A548, A552, A557, A562, A568, A579  
\empty@sfcnt .....  
.. x490, x491, x492, x506, x511, x563  
\ENC@cmd ..... 1012  
\enc@update ..... 1012,  
v257, v259, v275, v278, x147, x173  
\end@dblfloat ..... P205  
\end@float .....  
.. 770, 1040, P189, P227, P243, P383  
\endlnechar ..... 270  
\err@rel@i ..... y12, y99, y132, y136  
\error@fontshape ..... v482,  
v497, v522, v557, x107, x527, z332  
\escapechar ..... 265  
\et@xmaxfam ..... d22, d26, d30, d38  
\et@xmaxregs .....  
d29, d31, d32, d33, d34, d35, d36, d37  
\every@math@size .....  
.. 1016, 1019, v78, x235, x247  
\every@size ... 993, 1011, 1019, 1022  
\execute@size@function .....  
.. x362, x390, x404, x421  
\expand@font@defaults .. 203, A37,  
A148, A256, A279, A309, A330,  
A354, A365, A396, A397, A436,  
A438, A470, A472, A501, E26, E83  
\expand@font@defaults ..... A420  
\external@font .....  
.. x84, x87, x98, x102, x104, x391,  
x405, x467, x501, x569, x571, x573  
\extra@def ..... 992, y9, y84  
\extract@alph@from@version ...  
995, v597, v603, z162, z230, z260, z292  
\extract@default@composite .....  
.. s1062, s1069  
\extract@default@composite@o ...  
.. s1071, s1075  
\extract@default@composite@b ...  
.. s1073, s1077

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=lt hyphen.dtx, Z=ltfinal.dtx

```

\extract@font v378, x81
\extract@fontinfo x358, x365
\extract@rangefontinfo
..... x375, x382, x401, x434
\extract@sizefn x350, x372
\f@... 1011, 1047
\f@baselineskip s870, s1220,
..... v317, v324, v508, x121, x167,
..... x182, x201, x215, x226, x240
\f@depth P291, X345
\f@encoding . 995, 1011, 1012, d260,
..... d275, d283, s178, v251, v270, v273,
..... v274, v276, v326, v358, v363, v382,
..... v384, v386, v391, v393, v424, v486,
..... v518, v553, w439, w443, w652,
..... w656, x91, x128, x307, x517, z317
\f@family
..... 416, 524, 532, 992, v279, v287,
..... v299, v309, v320, v359, v363, v382,
..... v384, v386, v391, v393, v425, v504,
..... v529, v564, w439, w443, w652,
..... w656, x91, x128, z317, z348, A151,
..... A178, A179, A258, A259, A260,
..... A281, A282, A283, A319, A320,
..... A321, A339, A340, A341, A355,
..... A356, A357, A366, A367, A368,
..... A504, A517, A659, A677, A693,
..... E23, E27, E29, E31, E63, E66,
..... E80, E84, E86, E88, E93, E100,
..... E897, E900, E914, E923, E929, E1144
\f@linespread
..... 1012, v320, x120, x166, x183,
..... x184, x187, x195, x198, x209, x212
\f@series 434, 443, 444, 446,
..... 450, 459, 460, 521, 523, 529, 531,
..... 533, 540, q14, v279, v310, v321,
..... v360, v363, v505, v526, v561, w400,
..... w410, w419, w429, w463, w482,
..... w483, w652, w656, x125, x128,
..... x131, z349, A144, A160, A162,
..... A166, A167, A168, A189, A195,
..... A198, A200, A222, A520, A524,
..... A660, A678, A694, A739, E7, E613
\f@series@saved x125, x131
\f@shape 451, 459,
..... 540, v279, v289, v301, v311, v322,
..... v361, v363, v506, v524, v559, w439,
..... w443, w620, w627, w633, w643,
..... w650, w654, w657, w660, w669,
..... w676, w678, w713, x124, x128,
..... x130, z350, A661, A679, A695, A740
\f@shape@saved x124, x130
\f@size s180, s869, s1219,
..... v88, v317, v323, v362, v507, v546,
..... v580, v622, v623, v626, v627, x121,
..... x142, x167, x169, x180, x200, x215,
..... x218, x221, x226, x233, x240, x252,
..... x255, x261, x267, x284, x285, x288,
..... x293, x359, x366, x385, x387, x402,
..... x453, x455, x457, x473, x474, x479,
..... x493, x505, x510, x522, x530, x535,
..... x561, x575, A556, A565, E102, E931
\f@user@size .. x473, x478, x522, x535
\f@warn@break 1014
\famdef@ult A444
\filec@ntents 1024, T1098,
..... T1101, T1104, T1115, T1136,
..... T1229, T1232, T1235, T1246,
..... T1267, T1360, T1382, T1470, T1685
\filec@ntents@checkdir
..... T1121, T1123,
..... T1137, T1252, T1254, T1268, T1367
\filec@ntents@force
..... T1117, T1248, T1363
\filec@ntents@noheader
..... T1119, T1250, T1365
\filec@ntents@nosearch
..... T1120, T1251, T1366
\filec@ntents@opt . T1101, T1104,
..... T1106, T1232, T1235, T1237, T1362
\filec@ntents@OPTION 825
\filec@ntents@overwrite
..... T1118, T1249, T1364
\filec@ntents@where T1122, T1124,
..... T1149, T1253, T1255, T1280, T1368
\filename@area
..... a249, a255, a262, a268, a275, a281,
..... a288, r553, r568, r578, r610, r611,
..... r615, r640, r643, r660, r672, r674, T851
\filename@base a298,
..... a305, a318, r553, r568, r578, r610,
..... r612, r615, r640, r643, r667, r672, T852
\filename@dot a316, a322
\filename@dots a300, a302, a307
\filename@ext
..... 858, a297, a304, a314, a316, r554,
..... r569, r579, r606, r607, r610, r613,
..... r615, r636, r637, r640, r643, r668, T853
\filename@parse a113, a245,
..... r551, r566, r576, r605, r635, r665, T850
\filename@path .. a250, a251, a256,
..... a263, a264, a269, a276, a277, a282
\filename@simple a253, a266,
..... a279, a289, a293, a295, a310, a312
\finish@module@release c86, c90
\finph@nt
..... I104, I106, I110, I111, I119, I120

```

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr1.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspacel.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=lt page.dtx, T=lt class.dtx, U=ltkeys.dtx, V=lt filehook.dtx, W=lt shipout.dtx,  
X=lt output.dtx, Y=lt hyphen.dtx, Z=lt final.dtx

\finsm@sh .....  
 .... I134, I136, I140, I141, I145, I146  
\fix@penalty ..... D101  
\fixed@sfcnt ..... x565, x566, x567  
\fI@ShowFloat .. X1904, X1910, X1925  
\fI@trace .... X240, X267, X323,  
 X351, X358, X379, X426, X472,  
 X525, X540, X541, X542, X543,  
 X554, X555, X556, X557, X558,  
 X568, X781, X800, X819, X837,  
 X839, X978, X982, X994, X995,  
 X996, X997, X1003, X1006, X1014,  
 X1018, X1029, X1034, X1039,  
 X1040, X1041, X1042, X1049,  
 X1052, X1060, X1071, X1077,  
 X1082, X1087, X1093, X1094,  
 X1099, X1104, X1105, X1106,  
 X1114, X1118, X1123, X1127,  
 X1132, X1143, X1144, X1146,  
 X1164, X1173, X1179, X1188,  
 X1191, X1197, X1207, X1211,  
 X1221, X1227, X1233, X1239,  
 X1246, X1248, X1254, X1259,  
 X1261, X1263, X1271, X1276,  
 X1282, X1287, X1293, X1307,  
 X1308, X1311, X1332, X1341,  
 X1347, X1356, X1359, X1366,  
 X1376, X1380, X1392, X1398,  
 X1403, X1408, X1412, X1416,  
 X1417, X1424, X1429, X1433,  
 X1440, X1449, X1453, X1457,  
 X1458, X1462, X1463, X1473,  
 X1479, X1485, X1491, X1495,  
 X1501, X1503, X1511, X1516,  
 X1521, X1529, X1538, X1543,  
 X1548, X1550, X1555, X1557,  
 X1568, X1574, X1584, X1590,  
 X1594, X1595, X1600, X1601,  
 X1607, X1610, X1611, X1612,  
 X1619, X1620, X1621, X1629,  
 X1634, X1646, X1647, X1654,  
 X1657, X1665, X1669, X1673,  
 X1674, X1678, X1679, X1689,  
 X1695, X1705, X1711, X1715,  
 X1716, X1722, X1723, X1730,  
 X1733, X1734, X1735, X1743,  
 X1744, X1745, X1754, X1759,  
 X1772, X1774, X1781, X1784,  
 X1793, X1797, X1801, X1802,  
 X1806, X1807, X1859, X1864,  
 X1870, X1880, X1887, X1901,  
 X1902, X1906, X1917, X1929,  
 X2027, X2040, X2041, X2045,  
 X2048, X2050, X2053, X2056,  
 X2058, X2099, X2106, X2111,  
 X2117, X2122, X2126, X2132,  
 X2140, X2142, X2149, X2154,  
 X2159, X2161, X2167, X2169,  
 X2176, X2205, X2207, X2222,  
 X2224, X2238, X2263, X2267,  
 X2272, X2284, X2301, X2306, X2314  
\fI@tracemessage .....  
.... X1901, X1918, X1927, X1929  
\fI@traceval .... X1911, X1912,  
 X1913, X1914, X1917, X1926, X1929  
\fI@count .....  
.... b51, b52, b53, b62, b188,  
 b205, b210, b212, b213, b222, b230  
\fmtversion@topatch .....  
.... Z576, Z578, Z590, Z591, Z603, Z611  
\font@info ..... x99, x365, x434, x439  
\font@name ..... 992,  
 s179, s182, v86, v194, v196, v354,  
 v369, v545, v579, x84, x88, x90,  
 x105, x141, x144, x154, x168, x171,  
 x330, x331, x332, x333, x334, x339  
\font@submax 1007, x441, x470, x471,  
 H42, H44, H82, H84, Z285, Z287, Z296  
\fps@dbl ..... P34  
\freeze@math@version .... z154, z178  
\frozen@everydisplay .... v344, v350  
\frozen@everymath .... v344, v348  
\g@addto@macro .... 210,  
 f813, i60, w731, T133, T397, T1030,  
 T1046, T1047, W350, W356, W397  
\G@refundefined ..... 1029  
\G@refundefinedfalse .... 610, G5  
\G@refundefinedtrue .....  
.... 1029, G3, G12, R41, R68, R85  
\gen@sfcnt ..... x502, x503, x504  
\genb@sfcnt ... 1028, x507, x508, x509  
\genb@x ..... x510, x512  
\genb@y ..... x512  
\get@cdp ..... z479, z487, z520  
\get@external@font ... x83, x96, x536  
\getanddefine@fonts .....  
.... 492, v592, v610, x320, z59,  
 z87, z132, z159, z201, z227, z257,  
 z288, z386, z450, z484, z486, z503,  
 z626, z627, z659, z660, z1102, z1103  
\glb@currsize .... r41, r110, r168,  
 v341, x217, x252, x256, x262, x285  
\glb@settings . v342, x217, x264, x295  
\gobble@finish@module@release ...  
.... c106, c108, c167  
\gobble@font@spec ..... 1026  
\group@elt .....  
 z35, z384, z421, z422, z443, z447, z1134

**File Key:** a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\group@list ..... 1035, z388, z428, z441, z446, z447, z476, z702, z750, z811, z896, z899, z950, z953, z1019, z1022, z1089, z1140  
\h@false ..... I81  
\h@true ..... I82, I83  
\hb@xt@ 1023, 1033, b516, f29, s425, K44, K65, K83, K205, K489, K493, K494, K495, L37, M31, M43, M62, M74, M105, M117, M265, M299, M302, M305, M307, M314, M373, M452, M588, M598, M741, M797, O218, O241, O248, X630, X640, X689, X699, X1843, X2257, X2258, X2262, X2289, X2290, X2296, I197, I363, I425, I440, I452, I479, I510  
\hexnumber@ ..... z723, z731, z766, z774, z795, z805, z831, z839, z847, z856, z859, z868, z869, z908, z916, z962, z970, z989, z990, z1000, z1001, z1006, z1032, z1040, z1045, z1047, A645  
\hgl@ ..... b471, b472  
\hmode@bgroup ..... 1037, s67, s75, s327, s356, s390, s396, s427, s438, s445, s476, s483, s486, s488, s496, s512, s730, s760, s766, s798, s805, s833, s836, s893, s1253, D7, E625, E632  
\hmode@start@before@group ..... s68, s151, s153, s159, s184  
\if@afterindent ..... O124, O131  
\if@compatibility ..... T2, T611  
\if@endpe H214, H221, H273, H288, J138  
\if@eqnsw ..... I344, I410  
\if@fcolmade ..... X95, X264, X394, X403, X441, X451, X779, X799, X817, X846, X926, X2204, X2221, X2271, X2311  
\if@files w 1018, r5, r36, r105, r163, r297, r309, r330, r352, r364, r371, r383, H21, H48, H74, H88, O153, R4, R8, R39, R48, R56, R67, R84, T1147, T1164, T1278, T1297, W359  
\if@firstamp ..... L251  
\if@firstcolumn X95, X246, X279, X396, X444, X1815, X2235, X2280  
\if@font@series@context ..... 532, A509, A528, A536  
\if@forced@series ..... 444, 1049, w401, A143  
\if@ignore H4, H216, H223, H274, H289  
\if@includeinrelease ..... c70, c73, c129, f767  
\if@inlabel ..... J28, J65, J102, J160, J183, X161, X188  
\if@inmath ..... 1031  
\if@insert ..... X95, X1057, X1169, X1203, X1337, X1372, X1446, X1535, X1662, X1790  
\if@minipage p239, p257, p276, p311, H418, H440, J149, K323, L79, P20  
\if@mparswitch ..... X95, X1817  
\if@multiplelabels ..... G31  
\if@mypkg@draft ..... 841  
\if@negarg .. M157, M198, M212, M273  
\if@newlist H460, J29, J33, J69, J78, J106, J166, X599, X644, X657, X703  
\if@nmbrlist ..... J33, J201  
\if@no@font@opt ..... y16, y110, y129  
\if@nobreak ..... 1020, 1033, p120, p147, p278, p313, r202, r214, J167, J192, K286, K307, O47, O128, P180, P373, S29, S37, X165, X192, X335, X1148, X1314  
\if@noitemarg ..... J32, J199  
\if@noparitem ..... J30, J157  
\if@noparlist ..... J31, J114  
\if@noskipsec ..... p147, J58, K287, K308, O38, O40, O97, P374, X155, X182  
\if@ovb ..... M427, M495, M526, M551, M562, M575  
\if@ovhline ..... M459, M590  
\if@ovl ..... M427, M493, M522, M547, M592, M601  
\if@ovr ..... M427, M492, M521, M546, M589, M599  
\if@ovt ..... M427, M494, M525, M550, M567, M579  
\if@ovvline ..... M459, M565  
\if@partsw ..... r5, r301, r356  
\if@pboxsw ..... K278, K423  
\if@reversemargin ..... X101, X1820  
\if@reversemarginpar ..... X95  
\if@rjfield ..... L19, L33  
\if@skipping@module . c134, c146, c149  
\if@specialpage ... X95, X606, X664  
\if@tempswa ..... 1040, a81, a82, a83, b270, j9, r307, r362, s1542, v99, v687, z409, z464, z528, z609, z1133, H50, H90, H425, H446, R95, T1176, T1309, T1398, V293, V294, V316, V317, V337, V338, X990, X1026, X1626, X1751  
\if@test ..... X12, X13, X887, X906, X946, X968,

**File Key:** a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

X1032, X1116, X1125, X1274,  
 X1285, X1427, X1514, X1632, X1757  
 $\backslash\text{if@twocolumn}$  ..... r26, r95,  
 r152, P32, P210, P235, X95, X139,  
 X267, X278, X395, X443, X467,  
 X781, X837, X1814, X2206, X2223  
 $\backslash\text{if@twoside}$  ... X95, X138, X609, X667  
 $\backslash\text{ifdt@p}$  ..... I190, I192  
 $\backslash\text{IfFileExists@}$  ..... r416, r443  
 $\backslash\text{ifG@refundefined}$  ..... G3, G4, G5  
 $\backslash\text{ifh@}$  ..... I76, I114, I123  
 $\backslash\text{ifin@}$  ..... s1558, s1561, y50,  
 y52, z1, z22, z373, z475, z477, z538,  
 z551, z621, z623, z651, z703, z717,  
 z751, z763, z812, z828, z897, z900,  
 z921, z951, z954, z1017, z1020,  
 z1023, z1090, z1092, z1121, A210,  
 A214, A218, T234, T252, T483, T517  
 $\backslash\text{ifmath@fonts}$  ..... v204, x222  
 $\backslash\text{ifmaybe@ic}$  ..... 1040, D82, D91  
 $\backslash\text{ifnot@nil}$  ..... x343, x360, x381  
 $\backslash\text{ifrestore@version}$  ..... 1031  
 $\backslash\text{iftc@forced}$  ..... E871, E882, E1150  
 $\backslash\text{ifv@}$  ..... I75, I113, I122  
 $\backslash\text{ifx}$  ..... 270  
 $\backslash\text{in@}$  ... 447, 1040, s1556, s1559, y49,  
 y51, z1, z21, z372, z474, z476, z534,  
 z547, z620, z622, z649, z701, z712,  
 z749, z760, z810, z824, z895, z898,  
 z918, z949, z952, z1014, z1018,  
 z1021, z1088, z1091, z1119, A208,  
 A212, A216, T233, T250, T480, T516  
 $\backslash\text{in@@}$  ..... w489, w490, w492, w493, z5, z6, z7, z9  
 $\backslash\text{in@false}$  ..... z10  
 $\backslash\text{in@true}$  ..... z12  
 $\backslash\text{init@restore@glb@settings}$  .....  
 ..... x265, x268, x270  
 $\backslash\text{init@restore@version}$  .....  
 ..... 1031, z62, z91, z108, z123, z124  
 $\backslash\text{init@series@setup}$  .....  
 ..... w732, w737, A120, A206, A246  
 $\backslash\text{input@path}$  .... 11, 849, 857, 864,  
 1002, 1053, a112, a134, a136, a142,  
 a144, a150, a152, a157, a159, a169,  
 a236, r429, r450, r471, r498, r515, V276  
 $\backslash\text{insc@unt}$  .... b37, b51, b52, b53,  
 b62, b90, b91, b92, b94, b247, b248,  
 b249, b250, b251, b252, b263, b264,  
 b265, b266, b267, b271, b273, b292,  
 b293, b294, b295, b296, b297, X61  
 $\backslash\text{install@mathalphabet}$  .....  
 ..... v587, v604, v611, z392,  
 z395, z481, z482, z579, z631, z634,  
 z641, z656, z657, z664, z1104, z1106  
 $\backslash\text{is@range}$  ..... x376, x377  
 $\backslash\text{kernel@ifnextchar}$  .....  
 ..... c87, f81, f100, f150, f678, f693, T377  
 $\backslash\text{kernel@make@fragile}$  30, f375, f794,  
 f795, f796, f797, f798, f799, f800,  
 f801, f802, f803, f804, f805, f806,  
 f807, f808, f809, p24, p25, p26, p27,  
 p28, s172, s173, H394, H395, H396,  
 L160, L161, L162, M823, M824,  
 M825, M826, M827, M828, M829,  
 M830, M831, M832, M833, M834,  
 O23, O24, O25, O26, O27, S43,  
 S44, I90, I91, I92, I93, I166, I167, I168  
 $\backslash\text{l@ngrel@x}$  .... f74, f75, f76, f120, f167  
 $\backslash\text{l@nohyphenation}$  .....  
 ..... 1045, H422, H562, Z269  
 $\backslash\text{last@fontshape}$  v520, v538, v555, v572  
 $\backslash\text{latexrelease@postltcmd}$  ..... g2664  
 $\backslash\text{latexrelease@postltexpl}$  ..... e132  
 $\backslash\text{leavevemode@ifvmode}$  .....  
 .. p461, p462, p470, B624, B626,  
 B628, B630, I115, I141, I206, I226, I227  
 $\backslash\text{load@onefile@withoptions}$  .....  
 ..... 817, T855, T897, T1009  
 $\backslash\text{load@onefilewithoptions}$  .....  
 ..... 821, T808, T959, T1485  
 $\backslash\text{lower@bound}$  ..... x386, x387, x398  
 $\backslash\text{lst@vskip}$  ..... 303  
 $\backslash\text{ltx@sh@ft}$  ..... 1040,  
 b508, s390, s397, s476, s484, s760, s767  
 $\backslash\text{m@ne}$  ..... 273, b39  
 $\backslash\text{m@th}$  .....  
 ..... 1015, 1029, 1034, b488, b500, q13,  
 B337, B459, B461, B462, B465,  
 B506, B530, B533, B537, B540,  
 B547, B550, B557, B560, B642,  
 K278, K449, L181, O214, O237,  
 P400, P409, P416, P437, P444,  
 I68, I71, I106, I136, I154, I156,  
 I172, I191, I354, I440, I452, I479, I490  
 $\backslash\text{makeph@nt}$  ..... I101, I103  
 $\backslash\text{makesm@sh}$  ..... I131, I133  
 $\backslash\text{mandatory@arg}$  ..... x414, x501,  
 x505, x510, x517, x519, x524, x526,  
 x531, x533, x546, x562, x569, x571  
 $\backslash\text{math@bgroup}$  .... 491, v618, x306,  
 x312, z53, z81, z146, z175, z221,  
 z243, z251, z282, D130, D131, D138  
 $\backslash\text{math@egroup}$  ..... 993,  
 v618, x310, x311, D131, D132, D139  
 $\backslash\text{math@famname}$  ..... 992

**File Key:** a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrn.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfnctcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=lt hyphen.dtx, Z=ltfinal.dtx

\math@fonts .... v588, v593, x232,  
x336, z60, z89, z160, z228, z258, z290  
\math@fontsfalse ....  
.. q7, s302, s329, s360, s1255, v77,  
v206, v216, v239, E103, E673, E932  
\math@fontstrue .... v204, v630  
\math@group ..... 992  
\math@version ....  
.... 490, 992, v7, v336, v592,  
v596, v598, v599, v601, x230, z56,  
z59, z64, z65, z69, z84, z88, z93,  
z94, z98, z111, z112, z113, z126,  
z127, z128, z149, z151, z153, z154,  
z159, z163, z165, z167, z171, z224,  
z227, z231, z233, z235, z239, z254,  
z257, z261, z263, z265, z269, z285,  
z289, z293, z295, z297, z301, A617  
\math@version ..... 992  
\mathchar@type .. z795, z805, z856,  
z859, z868, z884, z989, z1000, z1063  
\mathph@nt ..... I99, I105  
\mathsm@sh ..... I129, I135  
\maybe@ic .... D63, D64, D83  
\maybe@ic@ ..... D83  
\maybe@icfalse ..... D97  
\maybe@ictrue ..... D87  
\maybe@load@fontshape ....  
... 358, s71, w476, w515, x127, A164  
\maybe@update@bfseries@defaults ..  
.... A38, A257, A266, A304  
\maybe@update@mdseries@defaults ..  
.... A39, A280, A289, A305  
\mb@b ..... K55, K66, K74, K84  
\mb@l ..... K55,  
K59, K65, K74, K78, K83, M137, M141  
\mb@r ..... K55,  
K59, K65, K74, K78, K83, M137, M141  
\mb@t ..... K56, K63, K75, K82  
\md@def@ult ..... A397  
\mddef@ult 1050, A198, A293, A294,  
A295, A334, A335, A336, A402, A443  
\mddefault@previous ..... 1053,  
A290, A292, A331, A333, B107, B117  
\mdseries@.. ..... 517, 523  
\mdseries@previous ..... 1053  
\mdseries@rm ....  
.... A111, A127, A128, A229,  
A237, A281, A293, A334, A339, A366  
\mdseries@sf ..... A112, A128,  
A238, A282, A294, A335, A340, A367  
\mdseries@tt ..... A113, A128,  
A239, A283, A295, A336, A341, A368  
\meaning ..... 257, 265, 268, 270, 271  
\merge@font@series .... 443-445,  
w408, w425, w426, w507, w509, x133  
\merge@font@series@ w428, w433, w510  
\merge@font@series@without@substitution  
..... 445, w460, w475, w512, x136  
\merge@font@series@without@substitution@  
..... w460, w513  
\merge@font@shape ....  
w626, w639, w640, w711, w716, x132  
\merge@font@shape@ . w642, w647, w717  
\merge@font@shape@without@substitution  
..... w666, w719, x135  
\merge@font@shape@without@substitution@  
..... w666, w720  
\mv@{version} ..... 491, 492  
\mv@{version}@frozen ..... 490  
\mv@{version}@reset ..... 491, 492  
\n@space .... B625, B627, B629,  
B631, B636, B637, B638, B639, B642  
\new@command ....  
.... f77, f78, f131, f165, f184, f239  
\new@environment .... f146, f147, f159  
\new@fontshape .... y2, y4, y22, y24  
\new@mathalphabet ... z532, z553, z564  
\new@mathgroup .... 993, 1025,  
b78, b80, b98, b100, d27, v14, z412  
\new@mathversion . z20, z361, z368, z371  
\new@module@skip .... c135, c147, c149  
\new@moduledate .. c79, c98, c101, c149  
\new@modulename ..... c93, c149  
\new@symbolfont ..... z413, z445  
\newcommand ..... 255, 257, 262  
\NewCommandCopy ..... 261  
\NewDocumentCommand ..... 255  
\newlinechar ..... 270  
\newmathalphabet@ ..... y14  
\newmathalphabet@@ ..... y109  
\newmathalphabet@@@ ..... y15, y109  
\nfss@catcodes ..... 423,  
1021, 1028, 1034, v19, v120, v387,  
v388, v395, v442, B40, B45, B135, X3  
\nfss@text ..... 995,  
1015, s315, s317, A648, D5, D122, G13  
\no@alphabet@error .....  
.... 993, 1007, v4, z391,  
z393, z569, z570, z584, z593, z679, z680  
\no@alphabet@help ..... 993, 1007  
\no@version@warning ..... 993  
\noaccents@ ..... v633, B129  
\noexpand ..... 271  
\non@alpherr v612, v614, z72, z101,  
z117, z174, z242, z272, z304, z1141

**File Key:** a=ltdirname.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfnrcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

\not@base . . . . . A720,  
   A724, A725, A726, A727, A728,  
   A729, A730, A731, A732, A733, A734  
 \not@math@alphabet . . . . . w527,  
   w532, w537, w683, w687, w690,  
   w693, w696, w699, w702, w705,  
   A5, A8, A11, A14, A17, A20, A23,  
   A26, A29, A255, A278, A308, A329,  
   A353, A364, A381, A384, A406,  
   A411, A416, A449, A454, A459,  
   A475, A478, A481, A484, A487, A597  
 \now@and@everyjob .. d215, d221, s1011  
 \o@ign .. . . . . b502, s390, s397, s476, s484, s760, s767  
 \on@line .. . . . . 298, h347, h730,  
   l8, l15, l214, A144, A147, H177,  
   H232, H249, H282, K150, T863, T998  
 \operator@font .. . . . . B643,  
   I3, I4, I5, I6, I7, I8, I9, I10, I11,  
   I12, I13, I14, I15, I16, I17, I18, I19,  
   I20, I21, I22, I23, I24, I25, I26, I27,  
   I28, I29, I30, I31, I32, I33, I34, I37, I40  
 \optional@arg .. . . . . x415, x494, x496, x568, x571  
 \outer@nobreak .. . . . . 1020, P181, P251, P255, P346, P364  
 \p@ .. . . . . b311  
 \p@... .. . . . . 1047, 1048  
 \p@enum .. . . . . 663  
 \p@equation .. . . . . I351, I488  
 \p@renwd .. . . . . 1003  
 \p@selectfont .. . . . . 993, x119  
 \par .. . . . . 158  
 \par@deathcycles .. . J56, J77, J79, J80  
 \patch@level .. . . . . 39, 1043, c1, c36, c41,  
   c43, c45, c49, c58, Z579, Z591, Z593  
 \patchcmd .. . . . . 255  
 \ph@nt .. . . . . I81, I82, I83, I97  
 \pickup@font s181, v195, v353, v547,  
   v581, x143, x170, x331, x333, x335  
 \pictur@ .. . . . . M21  
 \pkgcls@arg .. . . . . T1496, T1619  
 \pkgcls@candidate .. . T1483, T1498,  
   T1574, T1578, T1582, T1650, T1653  
 \pkgcls@debug .. . . . . T1473,  
   T1489, T1490, T1491, T1492,  
   T1493, T1550, T1551, T1552,  
   T1553, T1562, T1567, T1585,  
   T1594, T1609, T1643, T1644, T1645  
 \pkgcls@innerdate .. . . . .  
   . T1478, T1523, T1526, T1532, T1671  
 \pkgcls@mindate .. . . . .  
   . . . . . 834, T1503, T1512, T1528, T1533  
 \pkgcls@name .. . . . . T1495, T1538  
 \pkgcls@parse@date@arg . T1497, T1508  
 \pkgcls@parse@date@arg@ T1514, T1517  
 \pkgcls@parse@date@arg@version ..  
   . . . . . T1524, T1545  
 \pkgcls@releasedate .. . . . .  
   . . . . . T1483, T1579, T1583, T1654  
 \pkgcls@rollbackdate@error ..  
   . . . . . T1575, T1633, T1651  
 \pkgcls@show@selection .. . . . .  
   . . . . . T1602, T1607, T1657, T1662  
 \pkgcls@targetdate .. . . . . 834, T1478,  
   T1510, T1518, T1521, T1522,  
   T1526, T1534, T1535, T1548,  
   T1556, T1571, T1573, T1603,  
   T1614, T1616, T1641, T1647, T1649  
 \pkgcls@targetlabel .. . . . .  
   . . . . . 834, T1478, T1511, T1531, T1546,  
   T1558, T1590, T1623, T1661, T1664  
 \pkgcls@use@this@release .. T1559,  
   T1576, T1578, T1591, T1601, T1653  
 \pr@os .. . . . . I246, I254  
 \pr@ct .. . . . . I249, I255  
 \pr@ms .. . . . . I243, I244  
 \preload@sizes .. . . . . 992, y11, y94  
 \prepare@family@series@update .. . . . .  
   . . . . . 529, A117, A141, A243,  
   A407, A412, A417, A450, A455, A460  
 \pretocmd .. . . . . 254  
 \prim@os .. . . . . I240, I242, I254  
 \prime@os .. . . . . I241  
 \process@table .. . r40, r109, r167, z310  
 \protected .. . . . . 241  
 \protected@cmd .. . . . . 1025  
 \protected@edef .. . . . .  
   . . . . . 366, 510, 988, 1018, f253, t192,  
   A546, G43, G47, G55, G59, G66,  
   K380, K397, K414, O60, P472,  
   P490, P508, Z559, Z565, Z570, Z571  
 \protected@file@percent .. . . . .  
   . . . . . 619, 752, H122,  
   H129, H144, H152, H153, O165, O172  
 \protected@wlog T323, T325, T339, T353  
 \protected@write .. . . . . 1018,  
   1030, r201, r206, G33, O181, Q14, Q31  
 \protected@xdef .. . . . . 1018, 1021, f253,  
   O11, P453, P521, P537, T315, T334  
 \provide@command .. . . . . f178, f179  
 \ps@empty .. . . . . S10, Z144  
 \ps@plain .. . . . . S13  
 \q@curr@file .. . . . . 1052,  
   T1098, T1138, T1140, T1145,  
   T1171, T1269, T1271, T1276, T1304  
 \quote@@name .. . . . . r407, r423

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdbhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\quote@name ..... r407,  
     r420, r422, r499, r501, r510, T1269  
 \r@t ..... I66  
 \reenable@package@load .....  
     ..... 852, 867, 895, V478, W453  
 \reinstall@nfss@defs ..... w685,  
     w726, w730, w732, w736, w737, w740  
 \relax ..... 271  
 \rem@pt ..... v329  
 \remove@angles ..... x347, x370  
 \remove@nil ..... z36  
 \remove@star ..... x347, x353  
 \remove@tlig .....  
     ..... 1045, 1052, s1001, s1003,  
     s1005, s1029, s1034, s1081, s1082, s1083  
 \remove@to@nnil v328, x347, x373, x486  
 \renew@command .. f124, f125, f185, f193  
 \renew@environment ..... f152, f153  
 \requested@test@context .....  
     ..... 532, A503, A520, A524  
 \reserved@b ..... 506  
 \reserved@a .....  
     ..... 452, 532, 997, 1020, 1027,  
     a124, a128, a129, a198, a199, a202,  
     a220, a224, a246, a253, a256, a258,  
     a259, a266, a269, a271, a272, a279,  
     a282, a284, a334, a335, a336, b193,  
     c13, c19, c34, c178, c179, e22, e23,  
     e24, e25, e26, e47, e52, e89, e95,  
     e105, f117, f120, f133, f134, f135,  
     f137, f184, f185, f186, f192, f193,  
     f194, f195, f198, f218, f226, f230,  
     f288, f292, f326, f330, f354, f358,  
     f469, f477, f478, f526, f529, f544,  
     f546, f549, f558, f579, f586, f608,  
     f611, f675, f684, k33, k37, l234,  
     p409, p412, r211, r212, r234, r243,  
     r252, r305, r360, r430, r432, r437,  
     r439, r451, r453, r458, r460, r461,  
     r462, r472, r474, r479, r481, r496,  
     r502, r506, r513, r519, r523, r552,  
     r555, r556, r558, r567, r570, r577,  
     r580, r602, r603, r604, r608, r616,  
     r633, r634, r638, r644, r666, r670,  
     r674, s81, s82, s86, s89, s97, s107,  
     s110, s119, s138, s143, s995, s999,  
     s1043, s1045, s1046, s1048, s1050,  
     s1058, s1067, v30, v31, v32, v41,  
     v44, v47, v63, v66, v69, v105, v108,  
     v110, v147, v151, v389, v392, v519,  
     v520, v535, v538, v543, v554, v555,  
     v568, v572, v577, v604, v607, v608,  
     v616, w435, w436, w439, w440,  
     w455, w456, w468, w469, w648,  
     w649, w652, w653, w674, w675,  
     x196, x198, x200, x210, x212, x215,  
     x344, x345, x358, x359, y53, y57,  
     z479, z488, z490, z534, z537, z547,  
     z550, z648, z650, z712, z716, z760,  
     z762, z823, z826, z918, z920, z1014,  
     z1016, z1118, z1120, z1136, z1138,  
     z1139, z1144, A40, A41, A73, A74,  
     A174, A175, A516, A517, D47,  
     D48, D53, D54, D65, D68, D88,  
     D95, H120, H121, H175, H176,  
     H181, H230, H231, H235, H247,  
     H248, H252, H280, H281, H285,  
     H296, H297, K57, K58, K61, K76,  
     K77, K80, K145, K151, L241, L245,  
     L250, L269, L360, L361, M199,  
     M201, M205, M478, M479, M506,  
     M507, M535, M536, P29, P30,  
     P32, P33, P63, P67, P72, P74,  
     P76, P78, P83, P84, P132, P136,  
     P142, P145, P148, P151, T131,  
     T134, T136, T228, T236, T240,  
     T246, T254, T258, T385, T387,  
     T388, T389, T393, T408, T410,  
     T411, T412, T416, T591, T595,  
     T601, T605, T683, T684, T687,  
     T729, T733, T745, T746, T748,  
     T757, T761, T773, T774, T776,  
     T784, T788, T800, T801, T802,  
     T804, T813, T816, T817, T819,  
     T900, T949, T966, T1007, T1109,  
     T1110, T1112, T1240, T1241,  
     T1243, T1285, T1286, T1288,  
     T1292, T1500, T1505, T1557,  
     T1558, T1589, T1590, T1660,  
     T1661, T1685, T1687, V161, V167,  
     V168, V169, X37, X46, X48,  
     X50, X877, X897, X1997, X1999,  
     X2000, X2089, X2091, X2097,  
     X2100, Z212, Z229, Z230, Z231,  
     Z238, Z239, Z240, Z477, Z480,  
     Z511, Z517, Z518, Z529, Z530,  
     Z531, Z538, Z539, Z540, Z557,  
     Z558, Z559, Z560, Z563, Z564,  
     Z565, Z566, Z592, Z595, Z596,  
     Z613, I405, I406, I407, I408, I410  
 \reserved@b .. a125, a126, e48, e54,  
     f109, f111, f118, f135, f136, f227,  
     f228, f230, f289, f290, f292, f327,  
     f328, f330, f355, f356, f358, f527,  
     f529, f545, f549, f583, f586, f676,  
     f686, k33, k34, k37, p410, p411,  
     p418, r303, r305, r358, r360, r497,  
     r499, r501, r514, r516, r518, r602,

**File Key:** a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

r603, r604, r669, r675, s90, s97, s112, s119, s998, s999, s1044, s1045, s1067, s1076, s1078, v31, v32, v38, v95, v97, v150, v151, v605, v616, w454, w455, w457, y47, y54, y71, y73, z405, z407, z460, z462, z487, z488, z489, z524, z526, z605, z607, z652, z653, z654, z661, z821, z825, z827, A181, A186, A190, A191, A198, A199, D52, D53, D66, D68, D95, D96, L246, L248, L250, P43, P44, P112, P113, T229, T230, T231, T233, T248, T251, T385, T408, T737, T743, T746, T765, T771, T774, T792, T798, T802, T813, T820, T1154, T1155, T1158, T1159, T1194, T1195, T1197, T1224, T1287, T1288, T1291, T1292, T1327, T1328, T1330, T1356, T1416, T1417, T1419, T1446, X786, X789, X803, X806, X823, X826, Z215, Z217, Z221, Z483, Z485, Z489, Z558, Z564, Z613  
\reserved@c ..... 1034, a126, a131, f681, f684, f686, f689, r659, v96, v97, v606, v609, y48, y55, y61, y68, z33, z37, z406, z407, z461, z462, z525, z526, z606, z607, z629, z638, z653, z667, z908, z925, z934, z962, z973, z1031, z1044, z1046, A183, A186, A196, A197, A200, A201, D67, D69, D76, T814, T816, T817, T1139, T1144, T1145, T1163, T1171, T1177, T1199, T1207, T1270, T1275, T1276, T1296, T1304, T1310, T1332, T1339, T1387, T1388, T1389, T1399, T1421, T1428, T1456, Z219, Z224, Z232, Z477, Z487, Z508, Z509, Z510, Z512, Z513, Z514, Z515, Z516, Z524, Z532, Z615  
\reserved@d ..... 1050, a129, a132, e46, e51, f674, f683, r657, r659, y61, y68, y70, y74, z916, z925, z934, z970, z973, z1039, z1044, z1048, A188, A189, A194, A195, Z616  
\reserved@e ..... 1027, p57, p59, p69, p71, p100, p107, p115, y39, y45, y70, y73, y74, z34, z39, Z617  
\reserved@f ..... 1020, p58, p59, p70, p71, p115, s1543, s1544, s1545, s1546, s1548, s1555, v190, v192, v198, v199, x382, x393, x397, x401, x407, x410, x449, x486, x489, y27, y38, y45, y71, y73, Z618  
\reset@font ..... . 523, 994, 996, 1025, A219, A653, A684, A699, A714, G13, K376, K394, K411, P175, P371, P466, P485, P503, R40, S14, X616, X675  
\restglb@settings ..... x268, x278  
\restore@mathversion ..... z107, z110, z125, z133  
\restore@protect ..... f253  
\rlh@ ..... B464, B465  
\rm@def@ult ..... A397  
\rmdef@ult ..... 524, 532, A258, A281, A319, A339, A355, A366, A398, A439, E27, E84  
\robust@command@act ..... 94, 96, 97, 258, 260, 268, 1052, f410, f411, f413, f479, f503, i89  
\robust@command@act@chk@args ..... 94, 95, f435, f456  
\robust@command@act@do ..... 95, f421, f453  
\robust@command@act@end ..... 94, f418, f419, f431, f434, f454  
\robust@command@act@loop ..... 95, f415, f421, f451  
\robust@command@act@loop@aux ..... f421, f452  
\robust@command@chk@safe ..... 91, 100, 262, 1057, f303, f414, f435, f455, f560, f572, i136  
\s@fct@ ..... x426, x490  
\s@fct@alias ..... x552  
\s@fct@fixed ..... x565  
\s@fct@gen ..... x502  
\s@fct@genb ..... x507  
\s@fct@sgen ..... x502  
\s@fct@sgenb ..... x507  
\s@fct@sub ..... x514  
\s@fct@subf ..... x557  
\ saved@space@catcode ..... Z341, Z410  
\ scan@@fontshape ..... y7, y40, y43  
\ scan@fontshape ..... y6, y26, y37  
\ scantokens ..... 257, 268, 269  
\ scriptfont@name ..... x333, x338  
\ section ..... 256  
\ select@group ..... 490, v589, v608, z48, z346, z396, z534, z587, z596, z634, z666  
\ series@change@debug ..... A137, A144, A147, A158, A161, A165, A177, A185, A190, A196, A199, A201  
\ series@check@toks ..... w490, w492, w499  
\ series@drop@one@m ..... w496, w500, w518

**File Key:** a=ltdirname.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisenc.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

```

\series@maybe@drop@one@m v31, w483, w485,
..... w517, A168, A187, A193, A401, A402
\series@maybe@drop@one@m@x w486, w488
\seriesdefault@kernel 541, A220, A773
\set@mathdelimter z971, z1005
\set@color K88
\set@curr@file 817, 861,
..... 863–865, r225, r234, r261, r269,
..... r397, r415, T842, T1137, T1268, V267
\set@curr@file@aux V273, V279, V280
\set@curr@file@nosearch .. 1060, V267
\set@display@protect 1018,
..... f8, f16, f251, l7, l14, l34, l61, T326
\set@fontsize 1010, v317, v319, x121, x167, x177
\set@mathaccent z721, z729, z764, z772, z790
\set@mathchar z845, z855
\set@mathdelimter .. z922, z931, z983
\set@mathradical z354, z1041
\set@mathsymbol z829, z837, z858
\set@simple@size@args x348, x361, x368, x389, x403
\set@size@funct@args x351, x353, x411
\set@size@funct@args@ x411
\set@target@series v288, v300, w437, w441,
..... w444, w447, w470, w472, w481, w516
\set@typeset@protect 1018, 1032, f251, f270, L197, L235,
..... W84, W127, X603, X605, X661, X663
\SetMathAlphabet@ ... z541, z610, z619
\SetSymbolFont@ ... z431, z465, z473
\sf@def@ult A397
\sf@size q6, s302, v224, v243, v628, x328,
..... x332, E672, P409, P416, P437, P444
\sfdef@ult 524, A259, A282, A320, A340,
..... A356, A367, A399, A440, E29, E86
\sh@ft 1021, 1040, b506
\show@kernel@robust@command f566, f626
\ShowCommand 261
\sixt@n a74, b16, b64,
..... b66, b96, b97, b98, d30, v14, z84,
..... z285, z707, z709, z755, z757, z816,
..... z818, z864, z866, z904, z906, z912,
..... z914, z958, z960, z966, z968, z1027,
..... z1029, z1035, z1037, M278, M293,
..... M295, P62, P80, P131, P153,
..... X1005, X1051, X1190, X1358,
..... X1592, X1656, X1713, X1783,
..... X2043, X2052, X2108, X2124, X2157
\sixt@n 273
\size@update x146, x172, x185, x204, x206
\sizefn@info x352, x354, x362, x390, x404
\skip@ b41, b468, b470,
..... b471, b473, p83, p442, D105, D108
\sp@n L379
\split@name v357, v369, v483, v498, x519, x533
\ssf@size s328, s360,
..... s1254, v225, v244, v629, x328, x334
\string@makeletter 1053,
..... f701, T819, T851, T852, T853, V169
\strip@meaning 1003
\strip@prefix a114, a231,
..... a326, f228, f290, f328, f356, f698, v586
\strip@pt b510, v216, v222, v223, v224, v225,
..... v238, v242, v329, v628, v629, x180
\sub@sfcnt x514, x515, x516, x543
\subf@sfcnt x557, x558, x559
\subst@correction v85, v91
\subst@fontshape y8, y80
\subst@size x465
\sw@slant D91, D101
\t@st@ic 998, D90, D94
\target@meta@family@value A150, A175, A182, A184
\target@series@value 521, 528, 529, 1050,
..... A149, A157, A160, A162, A166,
..... A167, A168, A191, A197, A198, A200
\tc@check@accent E109,
..... E161, E163, E165, E167, E169,
..... E171, E173, E175, E177, E179,
..... E181, E183, E185, E187, E189,
..... E191, E938, E1014, E1016, E1018
\tc@check@symbol E109,
..... E211, E213, E215, E217, E219,
..... E221, E223, E225, E227, E229,
..... E231, E233, E235, E237, E239,
..... E241, E243, E245, E247, E249,
..... E251, E253, E255, E257, E259,
..... E261, E263, E265, E267, E269,
..... E271, E273, E275, E277, E279,
..... E281, E283, E285, E287, E289,
..... E291, E293, E295, E297, E299,
..... E301, E303, E305, E307, E310,
..... E312, E314, E316, E318, E320,
..... E322, E324, E326, E328, E330,
..... E332, E334, E336, E338, E340,
```

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

E342, E344, E346, E348, E350,  
 E354, E356, E358, E360, E362,  
 E364, E366, E938, E1008, E1010,  
 E1012, E1020, E1022, E1024,  
 E1026, E1028, E1030, E1032,  
 E1034, E1036, E1038, E1040,  
 E1042, E1044, E1046, E1048,  
 E1050, E1052, E1054, E1056,  
 E1058, E1060, E1062, E1064,  
 E1066, E1068, E1070, E1072,  
 E1074, E1076, E1078, E1080,  
 E1082, E1084, E1086, E1088,  
 E1090, E1092, E1094, E1096,  
 E1098, E1100, E1102, E1104,  
 E1106, E1108, E1110, E1112,  
 E1114, E1116, E1118, E1120, E1122  
 \tc@error ..... E110, E918, E939  
 \tc@errorwarn ..... E21, E76, E78, E825, E827,  
     E829, E830, E877, E878, E879, E912  
 \tc@fake@euro E97, E352, E926, E1007  
 \tc@forcedfalse ..... E871  
 \tc@forcedtrue ..... E876  
 \tc@oldstylesubst ..... E16, E20  
 \tc@subst ..... E77, E109, E911, E938  
 \tc@swap@accent ..... E112, E113  
 \test@font@series@context .....  
     532, A505, A515, A535  
 \test@next ..... 1026  
 \text@command ..... 1015, D8, D46  
 \textfont@name ..... x331, x337  
 \tf@size .....  
     996, v223, v243, v627, x328, x330  
 \thr@@ ..... 1023, b16, b530, b540,  
     b574, x58, x254, x260, x273, x280,  
     x287, x292, J232, J243, M287,  
     M288, M290, M291, M329, M355,  
     M388, M411, Z70, Z78, I363, I509  
 \toks@ 447, 1023, 1035, b41, c91, c103,  
     c105, c112, c116, c119, c124, f815,  
     f816, p408, p409, p414, v148, v152,  
     v154, v157, v221, v226, z6, z7, z382,  
     z386, z392, z395, z400, z446, z447,  
     z449, z450, z480, z482, z486, z503,  
     z506, z565, z577, z578, z579, z625,  
     z627, z633, z641, z645, z657, z660,  
     z663, z671, z673, z1094, z1096,  
     z1098, z1101, z1103, z1106, z1109,  
     z1141, z1142, T438, T439, T441,  
     T442, X2243, X2244, X2245, X2246  
 \try@load@font@shape ..... 1035  
 \try@load@fontshape .....  
     446, v372, v380,  
     v426, v531, w479, x520, z318, z335  
 \try@simple@size ..... x356, x481  
 \try@simples ..... x439, x445, x449  
 \try@size@range ..... x101, x356, x432  
 \try@size@substitution ..... x103, x436  
 \tryif@simple ..... x447, x448  
 \tryis@simple ..... x448  
 \tt@def@ult ..... A397  
 \ttdef@ult .....  
     524, A260, A283, A321, A341,  
     A357, A368, A400, A441, E31, E88  
 \tw@ ..... 273, 1007, 1015, b16  
 \two@digits ..... a89, a188, a189,  
     f2, x512, T1094, T1183, T1316, T1405  
 \type@restoreinfo ..... x202, x207  
 \undeclare@... ..... 895  
 \undeclare@file@substitution .....  
     852, V247  
 \unexpandable@noexpand ..... 1029  
 \unqu@tefilef@und ..... V143  
 \unquote@name . r401, r407, r424, V350  
 \unrestored@protected@xdef .....  
     f253, P458, P526, P542, P553, S25, S49  
 \update@series@target@value .....  
     A118, A152, A173, A244  
 \update@uclc@with@cyrillic .....  
     s1505, s1533, s1563, s1571  
 \upper@bound .. x383, x384, x385, x398  
 \use@mathgroup .....  
     993, v595, v613, v615, x299,  
     z63, z92, z547, z649, z652, z1119, z1143  
 \UTF@two@octets@noexpand ..... 1060  
 \UTFviii@four@octets Z396, Z401, Z407  
 \UTFviii@four@octets@@ ... Z396, Z407  
 \UTFviii@four@octets@combine . Z431  
 \UTFviii@four@octets@noexpand . Z437  
 \UTFviii@four@octets@string .. Z434  
 \UTFviii@invalid ..... Z335, Z428  
 \UTFviii@invalid@err Z393, Z398, Z404  
 \UTFviii@invalid@err@@ ... Z393, Z404  
 \UTFviii@three@octets Z395, Z400, Z406  
 \UTFviii@three@octets@@ .. Z395, Z406  
 \UTFviii@three@octets@combine . Z430  
 \UTFviii@three@octets@noexpand Z436  
 \UTFviii@three@octets@string . Z433  
 \UTFviii@two@octets Z394, Z399, Z405  
 \UTFviii@two@octets@@ ... Z394, Z405  
 \UTFviii@two@octets@combine .. Z429  
 \UTFviii@two@octets@noexpand .. Z435  
 \UTFviii@two@octets@string ... Z432  
 \UTFviii@undefined@err .....  
     Z392, Z397, Z403  
 \UTFviii@undefined@err@@ . Z392, Z403  
 \v@false ..... I82  
 \v@true ..... I81, I83

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=lxexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=lt hyphen.dtx, Z=ltfinal.dtx

```

\ver@<file>.<ext> 817
\verb@balance@group . H508, H510,
H523, H525, H538, H540, H546, H547
\verb@egroup H508, H511,
H523, H526, H538, H541, H547, H551
\verb@eol@error .. H548, H560, H570
\verb@font 998,
1027, H432, H453, H461, H561, H571
\verb@holig@list ... H576, H582
\verb@out 1034
\verb@elt
..... z18, z31, z32, z379, z380,
z429, z449, z540, z578, z670, z1099
\verb@list z16,
z21, z32, z372, z380, z434, z455,
z474, z545, z590, z620, z675, z1112
\vglo b468, b469
\voidb@x b311, b499, u18
\warn@rel@i y5,
y25, y29, y81, y85, y90, y95, y119, y140
\wrong@fontshape 992, v376, v513
\x@protect 1018,
f231, f242, f293, f331, f359, f530, f550
\xe@alloc@ Z42, Z52
\xe@alloc@intercharclass .. 1042, Z21
\xe@ch@ck Z43, Z47
\XXX@argdef 1005
\z@ 273, 1029, 1031, b311
\z@skip 303, 1031, b311
\zap@space r261, r281,
A546, R49, T229, T390, T413,
T425, T590, T706, T727, T745,
T756, T773, T783, T800, T1107, T1238
tex commands:
\tex_afterassignment:D
..... g2235, W49, W137
\tex_aftergroup:D ... 880, W57, W143
\tex_currentgrouplevel:D
..... W48, W56, W136, W142
\tex_deadcycles:D W91
\tex_endlinechar:D
.... g1712, g1714, g1715, g1721, i342
\tex_escapechar:D
167, 168, g989, g999, g1168, g1177,
g1615, g2102, g2128, i199, V212, V226
\tex_everypar:D
300–302, 1059, n18, n22, n26, n34,
n34, n74, n76, n86, n87, n138, n139
\tex_gdef:D 265, i189
\tex_hskip:D n60
\tex_indent:D n81
\tex_lastnodetype:D 304, n59
\tex_lowercase:D g1978, g2108
\tex_newlinechar:D i343
\tex_noindent:D 301, n24, n91
\tex_par:D 303, 304,
n16, n62, n69, n93, n134, n135, n136
\tex_parskip:D n23
\tex_setbox:D W50, W138
\tex_shipout:D .. W125, W369, W499
\tex_unskip:D 304, n55
\tex_vss:D W332
\tex_xdef:D 265, i195
\tex 995
text commands:
\tex_lowercase:n 1060, Z552
\tex_uppercase:n Z553
\textracute
s915, s1153, E234, E235, E721, E971
\textrascendercompwordmark
s854, E154, E685, E954
\textrasciacute
s965, s1114, E236, E237, E722, E995
\textrascibreve
s913, s1152, E238, E239, E723, E968
\textrascicaron
s914, s1151, E240, E241, E724, E969
\textrascicircum 1016,
1022, 1029, 1045, s286, s558, s1088
\textrasciidieresis
s953, s1101, E242, E243, E725, E985
\textrasciigrave
s904, s1082, E244, E245, E726, E966
\textrasciimacron
s960, s1109, E246, E247, E727, E990
\textrasciitilde 1016,
1022, 1029, 1045, s287, s559, s1093
\textrasteriskcentered 379, 1033, 1045,
1055, s267, s717, s864, s865, s1215,
t164, t170, E120, E577, E642, E961
\textrbackslash 1016, 1017,
1022, 1029, s268, s560, s718, s1087
\textrbaht s939,
s1156, E252, E253, E731, E1101, E1102
\textrbar 1016,
1022, 1029, s269, s561, s719, s1091
\textrbardbl 1038,
s270, s720, s919, s1170, t169, E127,
E365, E366, E578, E650, E797, E974
\textrbf 519, D19
\textrbigcircle 1039, s729, s892,
s1232, E254, E255, E732, E1053, E1054
\textrblank s861,
s1229, E345, E346, E785, E1023, E1024
\textrborn s905,
E256, E257, E387, E733, E1059, E1060
\textrbraceleft 1016,
1017, s271, s308, s562, s721, s1090

```

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntr1.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=lttoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

```

\textbraceright 1016,
 1017, s272, s309, s563, s722, s1092
\textbrokenbar s951, s1099, E128, E651, E983
\textbullet s273, s723,
 s921, s1179, E121, E579, E643, E976
\textcapitalcompwordmark s853, E153, E684, E953
\textcelsius s922, s1198,
 E129, E353, E354, E652, E791, E977
\textcent .. s947, s1095, E130, E653, E980
\textcentoldstyle .. 587, s924, E258,
 E259, E392, E395, E734, E1075, E1076
\textcircled 380, 581, 1017, 1021, s279, s283,
 s300, s301, s730, s893, E155, E156,
 E655, E671, E687, E858, E1123, E1125
\textcircledP s958,
 s1200, E260, E261, E735, E1117, E1118
\textcolonmonetary s926,
 s1191, E313, E314, E765, E1077, E1078
\textcommaabove 1042, s353,
 s355, s369, s370, s458, s459, s700, s701
\textcommabelow 1042, s324,
 s326, s332, s333, s703, s704, s705,
 s706, s707, s708, s709, s710, s711,
 s712, s1252, s1455, s1456, s1457, s1458
\textcompsubstdefault 1049, E34, E39, E91, E618, E916
\textcompwordmark .. 365, 1016, 1034,
 1039, 1049, s290, s291, s564, s1158
\textcopyleft s956,
 E262, E263, E368, E736, E1115, E1116
\textcopyright 1034, s283,
 s317, s954, s1102, E131, E654, E986
\textcurrency s949, s1097, E335,
 E336, E778, E853, E857, E1011, E1012
\textdagger 1017, s275, s312, s725, s917, s1177,
 t165, t171, E123, E581, E645, E972
\textdaggerdbl 1017, s274, s313, s724, s918, s1178,
 t166, t172, E122, E580, E644, E973
\textdblhyphen s876,
 E266, E267, E369, E738, E1025, E1026
\textdblhyphenchar s912,
 E264, E265, E370, E737, E1071, E1072
\textdegree . s961, s1110, E132, E656, E991
\textdied s907,
 E268, E269, E388, E739, E1063, E1064
\textdiscount s941,
 s1190, E270, E271, E740, E1105, E1106
\textdiv .. s978, s1135, E133, E657, E1005
\textdivorced s906,
 s1235, E272, E273, E741, E1061, E1062
\textdollar 1016, s255, s307,
 s438, s565, s798, s862, s1084, E114,
 E115, E623, E625, E959, E1131, E1133
\textdollaroldstyle . 587, s923, E274,
 E275, E393, E394, E742, E1073, E1074
\textdong s935,
 s1195, E315, E316, E766, E1095, E1096
\textdownarrow s903,
 s1210, E317, E318, E767, E1057, E1058
\texteightoldstyle s886,
 E214, E215, E384, E710, E1043, E1044
\textellipsis s296, s321, s1180
\textemdash 1016,
 s256, s407, s411, s566, s570, s785, s1162
\textendash 1016,
 s257, s408, s410, s567, s569, s786, s1161
\textestimated s942, s1206,
 E329, E330, E774, E856, E1009, E1010
\texteuro s976, s1196,
 E351, E352, E789, E854, E1006, E1007
\textexclamdown 1016, 1039,
 s258, s412, s414, s571, s787, s1094
\textfiguredash 1055, s410, s569, s1160, s1166
\textfiveoldstyle s883,
 E216, E217, E381, E711, E1037, E1038
\textfloatsep X728, X741, X2138, X2188, X2337
\textflorin s925, s1149, E333, E334, E777, E978
\textfont x337, I238
\textfouroldstyle s882,
 E218, E219, E380, E712, E1035, E1036
\textfraction 1035, X1950,
 X1953, X1978, X1981, X2130, X2331
\textfractionsolidus 1035,
 s877, s1187, E337, E338, E780, E962
\textgravedbl s916, s1154, E248, E249, E728, E970
\textgreater 1016, 1017, 1022,
 1025, 1029, s281, s572, s740, s1086
\textguarani s929,
 E276, E277, E391, E743, E1083, E1084
\textheight 1007, r22, r23, r91,
 r92, r148, r149, P257, P258, P261,
 P287, P301, W369, X78, X225,
 X226, X274, X399, X447, X474,
 X645, X704, X763, X815, Z142, Z143
\texthorizontalbar 1055, s411, s570, s1163, s1168
\texthyphen 1016, 1022,
 s260, s417, s574, s789

```

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr1.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\texthyphenchar . . . . .  
   ... 1016, 1022, s259, s416, s573, s788  
 \textindent . . . . . 1023  
 \textinterrobang . . . . . s933,  
   s1186, E349, E350, E787, E1091, E1092  
 \textinterrobangdown . . . . . s934,  
   s1236, E347, E348, E786, E1093, E1094  
 \textit . . . . . D21  
 \textlangle . . . . . s888,  
   s1227, E309, E310, E762, E1047, E1048  
 \textlbrace . . . . . 1016, 1017  
 \textlbrackdbl . . . . .  
   s900, E210, E211, E389, E707, E964  
 \textleaf . . . . . s908,  
   E278, E279, E373, E744, E1065, E1066  
 \textleftarrow . . . . . s859,  
   s1207, E319, E320, E768, E1019, E1020  
 \textlegacyasteriskcentered E589, E800  
 \textlegacybardbl . . . . . E589, E801  
 \textlegacybullet . . . . . E589, E802  
 \textlegacydagger . . . . . E589, E804  
 \textlegacydaggerdbl . . . . . E589, E803  
 \textlegacyparagraph . . . . . E589, E805  
 \textlegacyperiodcentered .. E589, E806  
 \textlegacysection . . . . . E589, E807  
 \textless . . . . . 1016, 1017, 1022,  
   1025, 1029, s280, s575, s739, s1085  
 \textlira . . . . . s931,  
   s1192, E321, E322, E769, E1087, E1088  
 \textlnot . . . s957, s1107, E134, E658, E988  
 \textlquill . . . . . s945,  
   s1188, E280, E281, E745, E1111, E1112  
 \textmacron . . . . . 1036  
 \textmarried . . . . . s909,  
   s1234, E282, E283, E746, E1067, E1068  
 \textmd . . . . . D19  
 \textmho . . . . . s891,  
   s1205, E284, E285, E747, E1051, E1052  
 \textminus . . . . .  
   . s889, s1211, E343, E344, E783, E963  
 \textmu s966, s1115, E341, E342, E782, E996  
 \textmusicalnote . . . . . s910,  
   s1233, E286, E287, E748, E1069, E1070  
 \textnaira . . . . . s928,  
   s1193, E288, E289, E749, E1081, E1082  
 \textnineoldstyle . . . . . s887,  
   E220, E221, E385, E713, E1045, E1046  
 \textnonbreakinghyphen . . . . .  
   . 1055, s409, s568, s1159, s1164  
 \textnormal . . . . . D15  
 \textnumero . . . . . s940,  
   s1199, E331, E332, E775, E1103, E1104  
 \textonekcentered 1039, s487, s698, s699  
 \textohm . . . . . s899,  
   s1204, E339, E340, E781, E855, E1008  
 \textonehalf s974, s1124, E135, E659, E1002  
 \textoneoldstyle . . . . . s879,  
   E222, E223, E377, E714, E1029, E1030  
 \textonequarter . . . . .  
   . s973, s1123, E136, E660, E1001  
 \textonesuperior . . . . . s970, s1118,  
   E137, E355, E356, E661, E792, E999  
 \textopenbullet . . . . . s943,  
   s1231, E290, E291, E750, E1107, E1108  
 \textordfeminine . . . . .  
   . s305, s955, s1103, E138, E662, E987  
 \textordmasculine . . . . .  
   s306, s971, s1119, E139, E664, E1000  
 \TextOrMath . . . . .  
   1041, t161, t164, t165, t166, t167,  
   t168, t169, t170, t171, t172, t177, t184  
 \textparagraph . . . . .  
   . 1017, s276, s310, s726, s967,  
   s1116, t168, E124, E582, E646, E997  
 \textperiodcentered . . . . . s277, s727,  
   s968, s1117, E125, E583, E647, E998  
 \textpermill . . . . . 1035  
 \textperthousand . . . . . 1035  
 \textpertenthousand . . . . . 584, 1035,  
   s492, s937, s1182, E306, E307,  
   E308, E759, E1097, E1098, E1135  
 \textperthousand . . . 1035, s490, s920,  
   s1181, E118, E119, E639, E975, E1134  
 \textpeso . . . . . s930,  
   s1197, E292, E293, E751, E1085, E1086  
 \textpilcrow . . . . . s938,  
   E294, E295, E386, E752, E1099, E1100  
 \textpm . . . s962, s1111, E140, E666, E992  
 \textquestiondown . . . . . 1016, 1039,  
   s261, s413, s415, s576, s790, s1126  
 \textquotedbl . . . . . 1016, s579, s1083  
 \textquotedblleft . . . . .  
   . 1016, s262, s418, s577, s791, s1174  
 \textquotedblright . . . . .  
   . 1016, s263, s419, s578, s792, s1175  
 \textquotingleft . . . . .  
   . 1016, s264, s420, s580, s793, s1171  
 \textquoteright . . . . .  
   . 1016, s265, s421, s581, s794, s1172  
 \textquotesingle . . . . .  
   . s863, s1081, E141, E667, E960  
 \textquotestraightbase . . . . .  
   . s855, E142, E371, E668, E955  
 \textquotestraightdblbase . . . . .  
   . s856, E143, E372, E669, E956  
 \texttriangle . . . . . s890,  
   s1228, E311, E312, E763, E1049, E1050

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfnctcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\textbrace ..... 1016, 1017  
 \textbrackdbl ..... . s901, E212, E213, E390, E708, E965  
 \textrecip ..... s932, s1201, E296, E297, E753, E1089, E1090  
 \textreferencemark ..... s969, s1185, E298, E299, E754, E1119, E1120  
 \textregistered 1017, 1029, 1039, s300, s301, s959, s1108, E144, E670, E989  
 \textrightarrow ..... s860, s1209, E323, E324, E770, E1021, E1022  
 \textrm ..... D15  
 \textrquill ..... s946, s1189, E300, E301, E755, E1113, E1114  
 \textsc ..... 1034, D21  
 \textsection ..... 1017, s278, s311, s582, s728, s952, s1100, t167, E126, E584, E585, E648, E984  
 \textservicemark ..... s944, s1202, E302, E303, E756, E1109, E1110  
 \textsevenoldstyle ..... s885, E224, E225, E383, E715, E1041, E1042  
 \textsf ..... D15  
 \textsixoldstyle ..... s884, E226, E227, E382, E716, E1039, E1040  
 \textsl ..... D21  
 \textssc ..... w536, D31, D39  
 \textssc ..... D25  
 \textsterling ..... 1016, s266, s319, s445, s583, s805, s948, s1096, E116, E117, E624, E632, E981, E1130, E1132  
 \textstyle ..... q15, B469, I63  
 \textsubscript ..... P419  
 \textsuperscript ..... 1023, 1024, s303, s305, s306, E663, E665, E679, P402  
 \textsurd ..... s972, s1226, E304, E305, E757, E1121, E1122  
 \textsw ..... w531, D30, D38  
 \textsw ..... D25  
 \TextSymbolUnavailable ... 1029, s3, s758  
 \textthreeoldstyle ..... s881, E228, E229, E379, E717, E1033, E1034  
 \textthreequarters ..... . s975, s1125, E146, E675, E1003  
 \textthreequartersemdash . s858, E145, E357, E358, E375, E674, E793, E958  
 \textthreesuperior ..... s964, s1113, E147, E359, E360, E676, E794, E994  
 \texttildebelow ..... . s911, s1155, E250, E251, E729, E967  
 \texttimes s977, s1129, E148, E677, E1004  
 \texttrademark ..... 1029, s303, s936, s1203, E149, E678, E979  
 \texttt ..... D15  
 \texttwelveudash ..... s857, E150, E361, E362, E374, E680, E795, E957  
 \texttwooldstyle ..... s880, E230, E231, E378, E718, E1031, E1032  
 \texttwosuperior ..... s963, s1112, E151, E363, E364, E681, E796, E993  
 \textulc ..... w526, D28, D29, D35, D37  
 \textunderline ..... D25  
 \textunderline ..... 1016  
 \textunderscore ..... 1016, 1025, 1034, s288, s315, s584, s1089  
 \textup ..... 449, D21  
 \textuparrow ..... s902, s1208, E325, E326, E771, E1055, E1056  
 \textvisiblespace ..... 1016, 1025, s292, s585, s1230  
 \textwidth ..... 1007, r24, r93, r150, K346, P266, X79, X144, X201, X218, X630, X640, X689, X699, X2257, X2289, Z143  
 \textwon ..... s927, s1194, E327, E328, E772, E1079, E1080  
 \textyen ..... s950, s1098, E152, E682, E982  
 \textzerooldstyle ..... s878, E232, E233, E376, E719, E1027, E1028  
 \TH ..... 1012, s535, s1131, Z569  
 \th ..... 1012, s586, s1137, Z569  
 \thanks ..... 743  
 \thanks ..... 1018, O10, O26  
 \the ..... 302, 996  
 \thebibliography (env.) ..... 782  
 \theenum ..... 663  
 \theequation ..... I339, I351, I427, I488  
 \thefootnote ..... P396, P521, P526, P546  
 \thempfn ..... K348, P453, P458, P537, P542, P545  
 \thempfootnote ..... K348, P398  
 \thepage ..... 891, r208, F6, G14, G34, O164, O171, O177, Q15, Q32, R43, S14, X244, X275, X1827  
 \Theta ..... B299  
 \theta ..... B275  
 \thetotalpages ..... 875, 891, W348, W431  
 \thicklines ..... M124  
 \thickmuskip ..... B646, I215, I217, I230  
 \thickspace ..... I201  
 \thinlines ..... M124, M816, M833  
 \thinmuskip ..... B644, I207, I209, I225, I231  
 \thinspace ..... 1050, p460, p466, p467, I176, I201, I238  
 \thispagestyle ..... S6  
 \tilde ..... B519  
 \time ..... a182, a186  
 \times ..... B396

**File Key:** a=ltdefns.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltcmd.dtx, g=ltcmdhooks.dtx, h=ltalloc.dtx, i=ltcntrl.dtx, j=ltlogos.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspacedtx, q=ltlength.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=ltxref.dtx, H=ltmiscen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=lthyphen.dtx, Z=ltfinal.dtx

```

\ttitle 743
\ttitle O6, O7, O21, O23, O31
tl commands:
 \c_empty_tl h66, h1114
 \c_novalue_tl 133,
 213, g456, g806, g857, g918, g1285,
 g1361, g1477, g1582, g1727, g1734,
 g1774, g1826, g1907, g1934, g2016
 \c_space_tl g24,
 g86, g103, g114, g115, g131, g139,
 g149, g152, g154, g156, g158, g166,
 g172, g222, g223, g1037, g1051,
 g1052, g1293, i282, n40, V550, V553
 \tl_clear:N g237,
 g282, g311, g378, g379, g576, g685,
 g695, g696, g697, g699, g842, g910,
 g1064, g1228, g1229, g1301, g1322,
 g1388, g1389, g1405, g1616, g1966, i193
 \tl_const:Nn g1251, g2240,
 h574, h575, h577, h578, h579,
 h582, h583, h585, i11, W309, W314
 \tl_count:N g329, g1990, V543
 \tl_count:n g515, g520, g761, g863, g917, g1104
 \tl_gclear:N 239, 240
 \tl_gclear_new:N h127
 \tl_gput_left:Nn U173
 \tl_gput_right:Nn 230, g1018, g1223, h293, n36, z188
 \tl_gremove_once:Nn h35, h35
 \tl_gset:Nn h105, h158,
 h255, h267, h283, h321, h430, h485,
 h488, h508, h511, i123, n13, z185, V51
 \tl_gset_eq:NN h64, h290, h301
 \tl_if_blank:nTF g511, g1371, g1788,
 g1829, g2627, g2628, g2629, h273
 \tl_if_empty:N 244
 \tl_if_empty:NTF g248,
 g249, g396, g626, g1235, h113,
 h204, h206, h981, h988, h1105, h1107
 \tl_if_empty:nTF g1785,
 g2103, g2129, g2164, g2365, g2460,
 h178, h191, h194, h473, h501, h565,
 h1282, i119, i226, i245, V40, V46,
 V58, V107, V114, V116, V118, V417
 \tl_if_empty_p:N h1233, h1234, W68, W114, W367
 \tl_if_empty_p:n h542
 \tl_if_eq:NNTF g271
 \tl_if_eq:nnTF g638, g876, g2108
 \tl_if_exist:N 241
 \tl_if_exist:NTF g1236, h81, h111,
 h1119, h1138, h1184, h1243, h1255
 \tl_if_exist_p:N h133, h134
 \tl_if_head_is_group:nTF i253
 \tl_if_head_is_N_type:nTF i250
 \tl_if_in:NnTF g1402
 \tl_if_in:nnTF 159, g527, g1368, g1394
 \tl_if_novalue:nTF ... g302, g321,
 g368, g846, g1858, g2619, g2620,
 g2621, g2622, g2623, g2624, h172
 \tl_if_single:nTF g1955, g2602
 \tl_if_single_token:nTF 178, g586, g2100, g2604, i236
 \tl_item:Nn g2060
 \tl_log:n h35, h37, h937
 \tl_map_function:nN g322, g512, g513, g861, g911, g2643
 \tl_map_inline:Nn g636
 \tl_map_inline:nn g1264, g1474
 \tl_new:N g11, g12,
 g13, g14, g17, g21, g28, g29, g30,
 g33, g37, g38, g40, g41, g46, g47,
 g1383, g1384, g1600, g1952, g2067,
 g2638, h25, h26, h27, h29, h32, h75,
 h84, h87, h96, h97, h156, h635,
 h760, h761, h762, i6, i8, i9, i10, n12,
 V8, V9, V10, V11, V59, W53, W203
 \tl_put_left:Nn g737, g1320
 \tl_put_right:Nn g297, g323, g466, g504, g518,
 g535, g540, g668, g752, g765, g792,
 g800, g814, g816, g823, g835, g851,
 g857, g926, g951, g1079, g1087,
 g1103, g1110, g1296, g1299, g1392,
 g1393, g1400, g1410, g1478, g1512,
 g1739, g1749, g1968, g2013, i165, i179
 \tl_replace_all:Nnn g1981
 \tl_rescan:nn i19, i19, i357, i383
 \tl_set:Nn g22, g84, g85, g238,
 g239, g240, g241, g269, g320, g771,
 g963, g981, g1143, g1292, g1327,
 g1365, g1471, g1498, g1505, g1518,
 g1524, g1530, g1536, g1542, g1548,
 g1554, g1560, g1591, g1613, g1635,
 g1948, g1949, g1975, g1999, g2030,
 g2034, g2060, g2210, g2211, h767,
 h787, h788, h793, h794, h809, h816,
 h817, h862, h869, i163, i176, i177,
 i180, i182, i203, i290, i295, i354,
 i380, V81, V82, V83, V84, W47,
 W135, W208, W234, W259, W286
 \tl_set_eq:NN g259, g265, g273, g316,
 g1321, g1404, g1477, h797, h820, i194
 \tl_show:N 167, g2083, g2090
 \tl_show:n g1191, g1201, h35, h36, h942, h1020
 \tl_tail:N g1669

```

**File Key:** a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfnctcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\tl\_to\_str:N ..... g1725, g1732  
 \tl\_to\_str:n ..... 157, 167, g66, g77, g182, g186, g391, g399, g412, g423, g427, g463, g557, g581, g589, g604, g617, g629, g648, g682, g1077, g1151, g1157, g1296, g1300, g1726, g1733, g1902, g1929, g2009, g2010, g2076, g2089, g2117, g2121, h222, h347, h971, h983, h1040, i112, i117, i352, i369, V577  
 \tl\_trim\_spaces:n ..... 266, g178, g349, g466, g1310, g1312, g1901, g1928, g2030, g2248, h251, i228, V41, V42  
 \tl\_trim\_spaces\_apply:nN ..... g586, g2094, h174  
 \tl\_use:N g1194, g1241, h928, z195, z199  
 \tmspace ..... 640, I201  
 \to ..... B440, B442  
 \today ..... a187, a191, a199, a202, O33  
 token commands:  
   \c\_space\_token ..... g2200  
   \token\_if\_active:NTF ..... g1741  
   \token\_if\_cs:NTF ..... 168, g1957  
   \token\_if\_eq\_catcode:NNTF g1422, i238  
   \token\_if\_eq\_charcode:NNTF ..... g344, g1657, g1689, g1700  
   \token\_if\_eq\_meaning:NNTF ..... g595, g598, g606, g1144, g2214, g2606, g2608, i184, i358, V455, V467  
   \token\_if\_macro:NTF ..... 261, g2166, i83  
   \token\_to\_meaning:N ..... i79, i355, i381  
   \token\_to\_str:N ..... 149, 167, g66, g74, g77, g349, g715, g958, g1072, g1193, g1203, g1435, g1581, g1742, g1901, g1928, g2104, g2117, g2121, g2130, g2169, g2176, g2177, g2178, g2180, g2249, g2512, g2513, g2526, g2527, g2568, g2569, g2582, g2583, h219, i78, i79, i101, i374, i377, V241  
 \toks ..... 302, b31, b63, b95, d36, n38, z576, z577, z587, z596, Z619  
 \toksdef ..... b46, b63, b95, d230  
 \tokszero ..... d230  
 \tolerance b317, v647, v692, v707, S58, S66  
 \top ..... B320  
 \topfigrule ..... X727, X2359  
 \topfraction ..... P273, X2325  
 \topmargin ..... X71, X624, X683  
 \topmark ..... X2243, X2252  
 \topsep ..... 652, 1019, J2, J59, I494  
 \topskip ..... 334, 1024, 1026, b388, r59, r127, r182, J1, X128  
 \totalheight ..... K33, K34, K35  
 totalpages ..... 875  
 trace commands:  
   trace\_stack\_levels ..... Z82  
 \tracefloats ..... X1929  
 \tracefloatsoff ..... X1929  
 \tracefloatvals ..... 957, 1057, X1929  
 \traceoff ..... 289  
 \traceon ..... 289  
 \tracingall ..... 289, b524  
 \tracingassigns ..... b542, b576, b593, b634  
 \tracingcommands ..... b343, b540, b558, b574, b583, b596, b620, b637  
 \tracingfonts ..... 993, 994, x17, x54, x58, x86, x118, x153, x194, x224, x238, x254, x260, x273, x280, x287, x292, x301, x314, x322, x325  
 \tracinggroups ..... b532, b567, b605, b645  
 \tracingifs ..... b533, b568, b604, b644  
 \tracinglostchars ..... b342, b525, b530, b553, b565, b584, b608, b628, b648  
 \tracingmacros ..... b337, b539, b557, b573, b585, b607, b627, b647  
 \tracingnesting ..... b535, b570, b602, b642  
 \tracingnone ..... b589  
 \tracingoff ..... x118, x322  
 \tracingon ..... 994, x119, x323  
 \tracingonline ..... b336, b519, b595, b619, b636, b666, v656, X1915  
 \tracingoutput ..... b341, b520, b599, b623, b640, b663  
 \tracingpages ..... b340, b529, b552, b564, b584, b609, b629, b649  
 \tracingparagraphs ..... b339, b531, b554, b566, b585, b606, b626, b646  
 \tracingrestores ..... b344, b541, b559, b575, b585, b594, b625, b635  
 \tracingscantokens ..... b534, b549, b569, b603, b617, b643  
 \tracingstacklevels ..... 976, b347, b350, b352, b353, b357, b361, b525, b537, b601  
 \tracingstats ..... b338, b528, b551, b563, b583, b610, b630, b650, Z2  
 \triangle ..... B322  
 \triangleleft ..... B358, B482  
 \triangleright ..... B359, B482  
 \TrimSpaces ..... g2639  
 trivlist (env.) ..... J89  
 \trivlist ..... 659, H351, H401, H403, H417, H439, J89, L78, N35, N37, I478  
 \tt ..... 998  
 \ttdefault ..... A12, A216, A400, A417, A441, A460, A488, B50  
 \ttfamily ..... 203, 529, A10, A11, A410, A458, A459, A486, A487, D17, H461

**File Key:** a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx, X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

ttfamily ..... [A420](#)  
 \ttsubstdefault .... [B20](#), [B32](#), [E32](#), [E89](#)  
 \twocolumn ..... [1032](#), [X199](#)  
 \twocolumn[...] ..... [1027](#)  
 \twocolumn[] ..... [334](#)  
 \typein ..... [80](#), [80](#)  
 \typein ..... [f31](#)  
 \typeout ..... [80](#)  
 \typeout ..... [100](#), [298](#), [446](#), [874](#),  
     [1018](#), [1029](#), [a76](#), [a119](#), [a175](#), [a200](#),  
     [a202](#), [a214](#), [a229](#), [a236](#), [a247](#), [a260](#),  
     [a273](#), [a286](#), [a324](#), [c21](#), [c38](#), [c43](#), [c48](#),  
     [f3](#), [f36](#), [f43](#), [f567](#), [f568](#), [f601](#), [f602](#),  
     [f607](#), [f74](#), [r200](#), [r584](#), [r585](#), [r591](#),  
     [r625](#), [r663](#), [r673](#), [r676](#), [v366](#), [w478](#),  
     [A138](#), [A563](#), [A744](#), [A754](#), [A764](#),  
     [B9](#), [B125](#), [Q8](#), [Q25](#), [T321](#), [T337](#),  
     [T349](#), [T1474](#), [T1673](#), [T1676](#), [W104](#),  
     [W115](#), [W147](#), [X1918](#), [X1930](#),  
     [Z262](#), [Z573](#), [Z580](#), [Z592](#), [Z593](#), [Z601](#)  
  
**U**  
 \u ..... [s237](#), [s386](#),  
     [s472](#), [s589](#), [s596](#), [s616](#), [s623](#), [s754](#),  
     [s1242](#), [s1317](#), [s1318](#), [s1333](#), [s1334](#),  
     [s1343](#), [s1344](#), [s1357](#), [s1358](#), [s1359](#),  
     [s1383](#), [s1384](#), [s1409](#), [s1410](#), [E163](#), [E194](#)  
 \uccode ..... [Z226](#),  
     [Z234](#), [Z241](#), [Z243](#), [Z246](#), [Z248](#),  
     [Z526](#), [Z534](#), [Z541](#), [Z543](#), [Z546](#), [Z548](#)  
 \Ucharcat ..... [A624](#)  
 \uchyph ..... [b366](#)  
 \ulcdefault ..... [w526](#), [w605](#)  
 \ulcshape ..... [448](#),  
     [w526](#), [w604](#), [w698](#), [w699](#), [A549](#), [D29](#)  
 \Umathcode .....  
     .a127, [d30](#), [p476](#), [B15](#), [B50](#), [B65](#),  
     [E159](#), [E367](#), [H467](#), [Z146](#), [Z337](#), [Z520](#)  
 \unboldmath ..... [A615](#)  
 \UndeclareTextCommand .....  
     .... [580](#), [1008](#), [1035](#), [s191](#), [E115](#),  
     [E117](#), [E119](#), [E308](#), [E585](#), [E1130](#),  
     [E1131](#), [E1132](#), [E1133](#), [E1134](#), [E1135](#)  
 \undefined .....  
     [1028](#), [a12](#), [a14](#), [a20](#), [a60](#), [t71](#), [t72](#),  
     [t128](#), [t129](#), [t130](#), [t131](#), [t132](#), [t133](#),  
     [B116](#), [B117](#), [Z154](#), [Z166](#), [Z167](#), [Z187](#)  
 \undefinedpagestyle ..... [S4](#), [S8](#)  
 \underbar ..... [b489](#), [f788](#), [f809](#)  
 \underbrace ..... [B540](#)  
 \underline ..... [667](#)  
 \underline ..... [b489](#), [K445](#), [K446](#)  
 \underscore ..... [1034](#)  
  
 \unexpanded ..... [263](#), [265](#),  
     [1057](#), [1058](#), [f584](#), [f595](#), [D47](#), [H191](#),  
     [H237](#), [H254](#), [T740](#), [T1502](#), [T1504](#)  
 \unhbox ..... [995](#)  
 \unhcopy ..... [b491](#), [L345](#), [M742](#), [M798](#)  
 \unicodedataline .....  
     .... [d143](#), [d146](#), [d160](#), [d161](#), [d162](#)  
 \UnicodeEncodingName ..... [385](#), [s981](#),  
     [s987](#), [s1038](#), [s1044](#), [s1048](#), [s1059](#),  
     [s1063](#), [s1079](#), [s1080](#), [E376](#), [E377](#),  
     [E378](#), [E379](#), [E380](#), [E381](#), [E382](#),  
     [E383](#), [E384](#), [E385](#), [E386](#), [E387](#),  
     [E388](#), [E389](#), [E390](#), [E391](#), [E392](#), [E393](#)  
 \UnicodeFontFile ..... [s1036](#)  
 \UnicodeFontName ..... [s1037](#)  
 \UnicodeFontTeXLigatures ..... [s993](#), [s1033](#)  
 \unicoderead .....  
     .. [d143](#), [d157](#), [d158](#), [d159](#), [d160](#), [d165](#)  
 \uninstall ..... [d838](#)  
 \unitlength .....  
     .. [872](#), [873](#), [889](#), [K53](#), [K64](#), [K73](#),  
     [K83](#), [M5](#), [M29](#), [M30](#), [M32](#), [M34](#),  
     [M42](#), [M43](#), [M44](#), [M45](#), [M60](#), [M63](#),  
     [M73](#), [M74](#), [M84](#), [M85](#), [M93](#), [M94](#),  
     [M107](#), [M108](#), [M119](#), [M164](#), [M176](#),  
     [M241](#), [M256](#), [M316](#), [M318](#), [M332](#),  
     [M340](#), [M342](#), [M357](#), [M375](#), [M377](#),  
     [M392](#), [M397](#), [M399](#), [M414](#), [M417](#),  
     [M422](#), [M480](#), [M481](#), [M508](#), [M509](#),  
     [M537](#), [M538](#), [M612](#), [M628](#), [M648](#),  
     [M654](#), [M690](#), [M691](#), [M693](#), [M694](#),  
     [M697](#), [M698](#), [M700](#), [M701](#), [M712](#),  
     [M713](#), [M715](#), [M716](#), [M718](#), [M719](#),  
     [M721](#), [M722](#), [M750](#), [M751](#), [M753](#),  
     [M754](#), [M757](#), [M758](#), [M760](#), [M761](#),  
     [M772](#), [M774](#), [M776](#), [M778](#), [W326](#), [Z135](#)  
 \unkern ..... [v675](#)  
 \unless ..... [d151](#), [d159](#), [d161](#)  
 \unlhd ..... [A732](#)  
 \unpenalty .....  
     .. [299](#), [v678](#), [v682](#), [D116](#), [H433](#), [H455](#)  
 \unrhdb ..... [A734](#)  
 \unsetattribute ..... [45](#)  
 \unsetattribute ..... [d82](#), [d240](#)  
 \unskip ..... [299](#), [995](#), [1047](#)  
 \unvcopy ..... [I180](#)  
 \Uparrow ..... [B584](#)  
 \uparrowarrow ..... [B578](#)  
 \upbracefill ..... [B543](#), [B560](#)  
 \updefault ..... [549](#), [1049](#), [w688](#),  
     [A21](#), [B94](#), [B101](#), [B103](#), [B111](#), [B113](#)  
 \Updownarrow ..... [B588](#)  
 \updownarrow ..... [B582](#)  
 \uplus ..... [B378](#)

**File Key:** a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

\uppercase ..... 1008  
 \upshape ..... 448, 450, 533, s442, s732,  
     s802, s895, w686, w687, A19, A20,  
     A549, A560, A593, A651, D24, E629  
 \Upsilon ..... B304  
 \upsilon ..... B286  
 use commands:  
     \use:N ..... e178,  
         g711, g1238, h878, h879, h919,  
         z202, U161, V443, V447, V449, V466  
     \use:n ..... 267, g136, g596,  
         g599, g759, g987, g1074, g1166,  
         g1210, g2603, g2605, g2610, h226,  
         h1068, h1194, i104, i169, i172, i200,  
         i206, i267, i300, i321, i349, i366, U119  
     \use:nn ..... h849, h1084, h1362, i211  
     \use:nnn ..... 176, 177, g2512, g2526, g2568, g2582  
     \use\_i:nn ..... g838, g1393, g2132, g2192  
     \use\_i:nnn ..... 228, g2110  
     \use\_i:nnnn ..... 267, i265  
     \use\_i\_delimit\_by\_q\_recursion\_-  
         stop:nw ..... i328  
     \use\_i\_delimit\_by\_q\_stop:nw .....  
         g358, g2170  
     \use\_ii:nn ..... 159, 228, g1377,  
         g1765, g1778, g1817, g1835, g1860,  
         g1893, g1920, g2131, g2614, i56, i138  
     \use\_ii:nnn ..... g1376, g1834, g2114, g2607  
     \use\_ii:nnnn ..... i270  
     \use\_ii\_i:nn ..... i210  
     \use\_ii\_iii:nnn .....  
         V216, V230, V242, V242  
     \use\_iii:nn ..... 228  
     \use\_iii:nnn ..... g1794,  
         g2109, g2112, g2191, g2609, V47  
     \use\_iii:nnnn ..... g1793  
     \use\_iv:nnnn ..... g344, g350  
     \use\_none:n ..... 241, g93, g600,  
         g1371, g1372, g1375, g1378, g1392,  
         g1419, g1556, g1829, g1830, g1833,  
         g1836, g2066, h7, h648, h709, h725,  
         h847, i272, i328, V131, V134, W7  
     \use\_none:nn .....  
         ... g1003, g1550, g1788, g1789,  
         g1792, g1795, h1040, h1066, z201  
     \use\_none:nnn ..... 159, g770, g1544, g1786, g2182  
     \use\_none:nnnn ..... g1538  
     \use\_none:nnnnn ..... g1532, h954  
     \use\_none:nnnnnn ..... g1526  
     \use\_none:nnnnnnn ..... g1520  
     \use\_none:nnnnnnnn ..... h648, h954  
     \use\_none\_delimit\_by\_q\_recursion\_-  
         stop:w ..... g272  
     \use\_none\_delimit\_by\_q\_stop:w .....  
         g2004, g2023, g2027  
 \usebox ..... K156  
 \usecounter ..... J225, J238  
 \usefont ..... 538, 1027, s1570,  
     v80, v282, v652, A709, E7, E613, H471  
 \UseHook .....  
     182, 183, 185, 186, 194, 196, 197,  
     208, 209, 254–256, 265, 270, 848,  
     850, 857, h1440, h1546, i212, i374,  
     i377, r315, r321, r326, x145, A263,  
     A274, A286, A296, A317, A324,  
     A337, A344, A403, A408, A413,  
     A418, A663, A680, H173, H185,  
     H188, H212, H215, H219, H222,  
     T913, T917, T941, T945, V170,  
     V171, V174, V175, W118, W168, W371  
 \UseLegacyTextSymbols ..... E576, E799  
 \UserName ..... 77  
 \UserName ..... e174, e192  
 \UseOneTimeHook .....  
     ... 182, 183, 185, 186, 196, 197,  
     1058, h1440, h1547, r15, r57, r70,  
     r316, r320, r325, H14, H18, H28,  
     H29, H31, T914, T918, T940, T944  
 \usepackage ..... 186, 187, 201,  
     848–850, 876, 1025, T611, T673, T1489  
 \UseRawInputEncoding .. Z367, Z423, Z470  
 \UseTextAccent .....  
     ... 361, 1016, s149, s150, s188,  
     E110, E111, E113, E156, E158,  
     E939, E1124, E1125, E1127, E1128  
 \UseTextSymbol ..... 361, 1016, 1034,  
     s150, s186, E109, E352, E938, E1007  
 \usetikzlibrary ..... 187  
 \ushape ..... 1035

## V

\v ..... s238, s387, s471, s592,  
     s593, s594, s598, s600, s603, s605,  
     s607, s613, s619, s620, s621, s625,  
     s627, s630, s632, s634, s640, s755,  
     s1247, s1327, s1328, s1329, s1330,  
     s1339, s1340, s1373, s1374, s1379,  
     s1380, s1391, s1392, s1399, s1400,  
     s1403, s1404, s1426, s1427, s1428,  
     s1429, s1430, s1431, s1432, s1433,  
     s1434, s1435, s1436, s1445, s1446,  
     s1447, s1448, s1451, s1452, E165, E195  
 \vadjust ..... 293,  
     295, p37, p59, p71, p100, p107,  
     p341, p357, p375, p391, P201, P223

**File Key:** a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
 S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
 X=ltoutput.dtx, Y=ltthyphen.dtx, Z=ltfinal.dtx

\valign . . . . .	E101, E930
\value . . . . .	397
\varbigtriangledown . . . . .	B362, B365
\varbigtriangleup . . . . .	B363, B364
\varepsilon . . . . .	q15, B291
\varphi . . . . .	B296
\varpi . . . . .	B293
\varrho . . . . .	B294
\varsigma . . . . .	B295
\vartheta . . . . .	B292
\vbadness . . . . .	887, b319, W210, W212, W261, W263, X2240
\vbox . . . . .	294, 303, 871, 886, 1022
vbox commands:	
\ vbox_set_to_ht:Nnn . . .	W213, W264
\ vbox_to_zero:n . . . . .	W324
\ vbox_unpack:N . . . . .	W224, W274
\vdash . . . . .	B404
\vdots . . . . .	B510
\vec . . . . .	B524
\vector . . . . .	l269, M233, M817, M834
\vee . . . . .	B367, B369
\verb . . . . .	996, 997, 1051, H465, H490, H503, H509, H518, H524, H534, H539, H552, H554
\verb* . . . . .	1047
\verb@ . . . . .	H459
\verb+ . . . . .	H459
\verb@+ . . . . .	H483
\verbvisible{space} . . . . .	1047, H465, H467, H474, H478, H490, H495
\Vert . . . . .	B571, B573
\vert . . . . .	B576
\vfil . . . . .	b478, E102, E105, E931, E934, M566, M578, W382, W394, X175, X194, X412, X459, X627, X686
\vfilneg . . . . .	b478
\vfuzz . . . . .	887, b374, S61, S68, W209, W211, W260, W262
\vglue . . . . .	b468
\vine . . . . .	L366
\phantom . . . . .	s497, s513, I75
\rule . . . . .	967, b472, p452, s293, s295, s502, s518, x190, B558, B559, B561, B562, K165, K167, K218, K225, K444, K488, L186, L219, L347, L366, M227, M299, M302, M324, M333, M350, M359, M383, M391, M406, M413, M565, M578, M726, M782, X1851, X2260, X2293
\vsize . . . . .	1030
\vskip . . . . .	293, 303, 315, 322, 995, 1005
\vspace . . . . .	p330, p400, p401, p402
\vsplit . . . . .	X382, X429, X2242
W	
\wedge . . . . .	B366, B368
\whatsit . . . . .	45, d194
\widehat . . . . .	B527
\widetilde . . . . .	B526
\widowpenalties . . . . .	b106
\widowpenalty . . . . .	b326, v665
\width . . . . .	1015, K30
\wlog . . . . .	804, 1005, a103, b40, b145, b239, b254, b284, b299, d6, d7, d8, d54, T351, Z46, Z626
\wp . . . . .	B312
\wr . . . . .	B382
\write . . . . .	29, 30, 202, 873, 883, 995, 1030
X	
\x . . . . .	c93, c96, v333, v334, Z316, Z318
\xdef . . . . .	992, 994
\XeTeXcharclass . . . . .	v640, Z25, Z33, Z40, Z53, Z59, Z68, Z75
\XeTeXcharclassCL . . . . .	Z159
\XeTeXcharclassCM . . . . .	Z163
\XeTeXcharclassEX . . . . .	Z160
\XeTeXcharclassID . . . . .	Z157
\XeTeXcharclassIS . . . . .	Z161
\XeTeXcharclassNS . . . . .	Z162
\XeTeXcharclassOP . . . . .	Z158
\XeTeXcharglyph . . . . .	s1034
\XeTeXdashbreakstate . . . . .	Z259
\XeTeXglyph . . . . .	s1034
\XeTeXintercharclasses . . . . .	Z153, Z186
\XeTeXinterchartoks . . . . .	Z154, Z168, Z169, Z170, Z171, Z172, Z173, Z174, Z175, Z176, Z177, Z178, Z179, Z180, Z181, Z182, Z187, Z192, Z193, Z194, Z195, Z196, Z197, Z198, Z199, Z200, Z201, Z202, Z203, Z204, Z205, Z206
\XeTeXmathcode . . . . .	Z147, Z521
\XeTeXrevision . . . . .	Z27
\XeTeXuseglyphmetrics . . . . .	Z256, Z258
\XeTeXversion . . . . .	A623, Z27
\Xi . . . . .	B301
\xi . . . . .	B281
\xpt . . . . .	992, 998
\xtxHanGlue . . . . .	.. Z166, Z190, Z198, Z199, Z200, Z201, Z202, Z203, Z204, Z205, Z206
\xtxHanSpace . . . . .	Z167, Z191, Z192, Z193, Z194, Z195, Z196, Z197
Y	
\y . . . . .	c94, c96

**File Key:** a=ltdirname.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx

\year . a188, c11, c14, T1183, T1316, T1405 \z ..... Z229, Z518, Z529  
Z \zeta ..... B273  
\z ..... Z238, Z517, Z538

**File Key:** a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,  
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,  
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,  
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,  
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,  
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,  
S=ltpage.dtx, T=ltclass.dtx, U=ltkeys.dtx, V=ltfilehook.dtx, W=ltshipout.dtx,  
X=ltoutput.dtx, Y=ltyphephen.dtx, Z=ltfinal.dtx