

The L^AT_EX 2 _{ε} Sources

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

2021-11-15 Patch level 1

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	2
2.2	Some bits of 2e	4
3	texsys.cfg	5
3.1	texsys.cfg	5
3.2	UNIX (web2c)	6
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 (???)	7
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	10

6	Filename Parsing	11
7	T_EX Versions	13
8	ltxcheck.tex	13
b	ltplain.dtx	14
1	Plain T_EX	14
c	ltvers.dtx	36
1	Version Identification	36
1.1	Declaring an all-new module	39
d	ltluatex.dtx	41
1	Overview	41
2	Core T_EX functionality	41
3	Plain T_EX interface	42
4	Lua functionality	42
4.1	Allocators in Lua	42
4.2	Lua access to T _E X register numbers	43
4.3	Module utilities	44
4.4	Callback management	44
5	Implementation	45
5.1	Minimum LuaT _E X version	45
5.2	Older L ^A T _E X/Plain T _E X setup	45
5.2.1	Fixes to <i>etex.src/etex.sty</i>	46
5.2.2	luatex specific settings	46
5.3	Attributes	47
5.4	Category code tables	47
5.5	Named Lua functions	49
5.6	Custom whatsis	50
5.7	Lua bytecode registers	50
5.8	Lua chunk registers	50
5.9	Lua loader	50
5.10	Lua module preliminaries	52
5.11	Lua module utilities	52
5.11.1	Module tracking	52
5.11.2	Module messages	53
5.12	Accessing register numbers from Lua	54
5.13	Attribute allocation	55
5.14	Custom whatsit allocation	56

5.15	Bytecode register allocation	56
5.16	Lua chunk name allocation	57
5.17	Lua function allocation	57
5.18	Lua callback management	57
5.18.1	Housekeeping	57
5.18.2	Handlers	60
5.18.3	Public functions for callback management	63
e	ltexpl.dtx	68
1	expl3-dependent code	68
1.1	Loader	68
1.2	Using expl3 code	71
f	ltdefns.dtx	73
1	Definitions	73
1.1	Initex initializations	73
1.2	Saved versions of <code>T_EX</code> primitives	73
1.3	Command definitions	74
1.4	Robust commands and <code>protect</code>	83
1.5	Acting on robust commands	89
1.5.1	Copying robust commands	91
1.5.2	Showing robust commands	92
1.5.3	Commands defined with <code>\DeclareRobustCommand</code>	93
1.5.4	Commands defined with <code>\newcommand</code> (with optional argument) .	95
1.6	Internal defining commands	96
2	Discretionary Hyphenation	100
g	ltcmd.dtx	103
1	Creating document commands	103
1.1	Variables and constants	103
1.2	Declaring commands and environments	106
1.3	Structure of <code>xparse</code> commands	110
1.4	Normalizing the argument specifications	115
1.5	Preparing the signature: general mechanism	123
1.6	Setting up a standard signature	124
1.7	Setting up expandable types	129
1.7.1	Copying a command and its internal structure	132
1.7.2	Showing the definition of a command	137
1.8	Grabbing arguments	141
1.9	Grabbing arguments expandably	153
1.10	Argument processors	158
1.11	Access to the argument specification	160
1.12	Utilities	162
1.13	Messages	166

1.14	User functions	171
------	--------------------------	-----

h	lthooks.dtx	176
1	Introduction	176
2	Package writer interface	176
2.1	L ^A T _E X 2 _ε interfaces	176
2.1.1	Declaring hooks	176
2.1.2	Special declarations for generic hooks	177
2.1.3	Using hooks in code	177
2.1.4	Updating code for hooks	178
2.1.5	Hook names and default labels	180
2.1.6	The top-level label	182
2.1.7	Defining relations between hook code	183
2.1.8	Querying hooks	184
2.1.9	Displaying hook code	185
2.1.10	Debugging hook code	186
2.2	L3 programming layer (<code>expl3</code>) interfaces	186
2.3	On the order of hook code execution	188
2.4	The use of “reversed” hooks	190
2.5	Difference between “normal” and “one-time” hooks	191
2.6	Generic hooks provided by packages	192
2.7	Private L ^A T _E X kernel hooks	192
2.8	Legacy L ^A T _E X 2 _ε interfaces	193
3	L^AT_EX 2_ε commands and environments augmented by hooks	194
3.1	Generic hooks	194
3.1.1	Generic hooks for all environments	194
3.1.2	Generic hooks for commands	196
3.1.3	Generic hooks provided by file loading operations	196
3.2	Hooks provided by <code>\begin{document}</code>	196
3.3	Hooks provided by <code>\end{document}</code>	197
3.4	Hooks provided by <code>\shipout</code> operations	198
3.5	Hooks provided for paragraphs	198
3.6	Hooks provided in NFSS commands	198
4	The Implementation	199
4.1	Debugging	199
4.2	Borrowing from internals of other kernel modules	200
4.3	Declarations	200
4.4	Providing new hooks	202
4.4.1	The data structures of a hook	202
4.4.2	On the existence of hooks	203
4.4.3	Setting hooks up	204
4.4.4	Disabling and providing hooks	207
4.5	Parsing a label	208
4.6	Adding or removing hook code	212
4.7	Setting rules for hooks code	221

4.8	Specifying code for next invocation	235
4.9	Using the hook	236
4.10	Querying a hook	239
4.11	Messages	242
4.12	L ^A T _E X 2 _{ε} package interface commands	244
4.13	Deprecated that needs cleanup at some point	247
4.14	Internal commands needed elsewhere	248
i	ltcmdhooks.dtx	250
1	Introduction	250
2	Restrictions and Operational details	251
2.1	Patching	251
2.1.1	Timing	251
2.2	Commands that look ahead	251
3	Package Author Interface	252
4	The Implementation	253
4.1	Execution plan	253
4.2	Variables	254
4.3	Variants	255
4.4	Patching or delaying	255
4.5	Patching commands	256
4.5.1	Patching by expansion and redefinition	257
4.5.2	Patching by retokenization	264
4.6	Messages	268
j	ltalloc.dtx	269
1	Counters	269
k	ltcntrl.dtx	271
1	Program control structure	271
l	lterror.dtx	275
1	Error handling and tracing	275
1.1	General commands	275
1.2	Specific errors	281
1.3	Tracing	285
m	ltpar.dtx	286

1	Paragraphs	286
1.1	Implementation	286
n	ltpara.dtx	288
1	Introduction	288
1.1	The default processing done by the engine	288
2	The new mechanism implemented for L^AT_EX	290
2.1	The provided hooks	291
2.2	Altered and newly provided commands	292
2.3	Examples	293
2.3.1	Testing the mechanism	293
2.3.2	Mark the first paragraph of each <code>itemize</code>	295
2.4	Some technical notes	295
2.4.1	Glue items between paragraphs (found with <code>fancypar</code>)	295
3	The Implementation	296
3.1	Providing hooks for paragraphs	296
3.2	The error messages	302
o	ltspace.dtx	304
1	Spacing	304
1.1	User Commands	304
1.2	Chris' comments	304
1.3	Some immediate actions	306
1.4	The code	307
1.5	Vertical spacing	314
1.6	Horizontal space (and breaks)	319
p	ltlogos.dtx	323
1	Logos	323
q	ltfiles.dtx	324
1	File Handling	324
1.1	Safe Input Macros	336
1.2	Listing files	343
r	ltoutenc.dtx	345

1	Font encodings	345
1.1	Removing encoding-specific commands	347
1.2	The order of declarations	348
1.3	Docstrip modules	348
1.4	Definitions for the kernel	349
1.4.1	Declaration commands	349
1.4.2	Hyphenation	357
1.4.3	Miscellania	357
1.4.4	Default encodings	357
1.4.5	Math material	360
1.5	Definitions for the OT1 encoding	361
1.6	Definitions for the T1 encoding	363
1.7	Definitions for the OMS encoding	369
1.8	Definitions for the OML encoding	370
1.9	Definitions for the OT4 encoding	370
1.10	Definitions for the TS1 encoding	372
1.11	Definitions for the TU encoding	377
2	Package files	387
2.1	The fontenc package	388
s	ltcounts.dtx	391
1	Counters and Lengths	391
1.1	Environment Counter Macros	391
t	ltlength.dtx	399
1	Lengths	399
u	ltfssbas.dtx	401
1	Preliminary macros	401
2	Macros for setting up the tables	402
3	Selecting a new font	409
3.1	Macros for the user	409
3.2	Macros for loading fonts	413
4	Assigning math fonts to <i>versions</i>	421
v	ltfssaxes.dtx	428

1	Changing the font series	428
1.1	The series lookup table	428
1.2	Mapping rules for series changes	429
1.3	Changing to a new series	437
2	Changing the shape	442
2.1	Mapping rules for shape combinations	443
2.2	Changing to a new shape	445
3	Make sure we win ...	447
w	ltfsstrc.dtx	449
1	Introduction	449
2	A driver for this document	449
3	The Implementation	450
4	Handling Options	450
5	Macros common to <code>fam.tex</code> and <code>tracefnt.sty</code>	452
5.1	General font loading	452
5.2	Math fonts setup	458
5.2.1	Outline of algorithm for math font sizes	458
5.2.2	Code for math font size setting	459
5.2.3	Other code for math	460
6	Scaled font extraction	462
6.1	Sizefunctions	470
x	ltfsscmp.dtx	474
y	ltfssdcl.dtx	479
1	Interface Commands	479
z	ltfssini.dtx	509
1	NFSS Initialization	509
1.1	Providing math <i>versions</i>	509
2	Custom series settings for main document families	510
3	Supporting nested emphasis	527
3.1	Legacy	531
3.2	Miscellaneous	531

A	fontdef.dtx	537
1	Introduction	537
2	Customization	537
3	The <code>docstrip</code> modules	538
4	A driver for this document	538
5	The <code>fonttext.ltx</code> file	538
5.1	Encodings	539
5.2	Defaults	541
6	The <code>fontmath.ltx</code> file	543
6.1	The font encodings used	543
6.1.1	Symbolfont and Alphabet declarations	543
6.2	Math font sizes	544
6.3	The math symbol assignments	545
6.3.1	The letters	545
6.3.2	The digits	546
6.3.3	Punctuation, brace, etc. keys	546
6.3.4	Delimitercodes for characters	546
6.4	Symbols accessed via control sequences	547
6.4.1	Greek letters	547
6.4.2	Ordinary symbols	548
6.4.3	Large Operators	548
6.4.4	Binary symbols	549
6.4.5	Relations	550
6.4.6	Arrows	551
6.4.7	Punctuation symbols	552
6.4.8	Math accents	553
6.4.9	Radicals	553
6.4.10	Over and under something, etc	553
6.4.11	Delimiters	554
6.5	Math versions of text commands	555
6.6	Other special functions and parameters	555
6.6.1	Biggggg	555
6.6.2	The log-like functions	556
6.6.3	Parameters	556
7	Default cfg files	556
B	preload.dtx	558
1	Overview	558
2	Customization	558
3	Module switches for the <code>DOCSTRIP</code> program	558

4	A driver for this document	559
5	The code	559
C	ltfntcmd.dtx	561
1	Introduction	561
2	The implementation	563
3	Initialization	569
D	lttextcomp.dtx	570
1	Sub-encodings	574
1.1	Sub-encoding 1 (drop symbols not working in Latin Modern)	575
1.2	Sub-encoding 2 (majority of new OTF fonts via autoinst)	575
1.3	Sub-encoding 3	579
1.4	Sub-encoding 4	579
1.5	Sub-encoding 5 (most older PS fonts)	579
1.6	Sub-encoding 6	580
1.7	Sub-encoding 7	580
1.8	Sub-encoding 8	580
1.9	Sub-encoding 9 (most missing)	580
2	Unicode engine specials	580
3	Font family sub-encodings setup	581
4	Legacy symbol support for lists and footnote symbols	585
5	The textcomp package	590
5.1	The old textcomp package code	591
5.1.1	Supporting oldstyle digits	599
5.1.2	Subset encoding defaults	600
E	ltpageno.dtx	602
1	Page Numbering	602
F	ltxref.dtx	603
1	Cross Referencing	603
1.1	Cross Referencing	603
G	ltmiscen.dtx	608

1	Miscellaneous Environments	608
1.1	Environments	608
1.2	Center, Flushright, Flushleft	620
1.3	Verbatim	623
H	ltmath.dtx	629
1	Math setup	629
1.1	Math commands based on plain TeX	629
1.1.1	The log-like functions	629
1.1.2	Biggggg	630
1.1.3	The UNSORTED Rest	630
1.2	Math Environments	636
1.3	External options to the standard document classes	641
1.3.1	Left equation numbering	641
1.3.2	Flush left equations	641
I	ltlists.dtx	644
1	List, and related environments	644
1.1	List and Trivlist	645
1.2	Vertical Spacing (skips)	646
1.3	Penalties	646
1.4	Horizontal Spacing (dimens)	646
1.5	Default Values	646
1.6	Itemize and Enumerate	657
J	ltboxes.dtx	660
1	L^AT_EX Box commands	660
1.1	Some low-level constructs	675
K	lttab.dtx	676
1	Tabbing, Tabular and Array Environments	676
1.1	tabbing	676
1.2	array and tabular environments	685
L	ltpictur.dtx	701
1	Picture Mode	701
1.1	Curves	728
M	ltthm.dtx	733

1	Theorem Environments	733
N	ltsect.dtx	737
1	Sectioning Commands	737
1.1	The Title	737
1.2	Sectioning	738
1.2.1	Initializations	745
1.3	Table of Contents etc.	745
1.3.1	Convention	745
1.3.2	Commands	745
O	ltfloat.dtx	750
1	Floats	750
1.1	Floating Environments	750
1.2	Footnotes	764
P	ltidxglo.dtx	773
1	Index and Glossary Generation	773
Q	ltbibl.dtx	776
1	Bibliography Generation	776
1.1	Default definitions	780
R	ltpage.dtx	781
1	Page styles and related commands	781
1.1	Page Style Commands	781
1.2	How a page style makes running heads and feet	781
1.3	marking conventions	781
S	ltclass.dtx	785
1	Introduction	785
2	User interface	785
2.1	Option processing	786

3	Class and Package interface	787
3.1	Class name and version	787
3.2	Package name and version	787
3.3	Requiring other packages	787
3.4	Declaring new options	788
3.5	Safe Input Macros	789
4	Implementation	789
4.1	Hooks	816
4.2	Providing shipment	818
5	Package/class rollback mechanism	826
6	After Preamble	834
T	ltfilehook.dtx	835
1	Introduction	835
1.1	Provided hooks	835
1.2	General hooks for file reading	835
1.3	Hooks for package and class files	836
1.4	Hooks for <code>\include</code> files	837
1.5	High-level interfaces for L ^A T _E X	838
1.6	Internal interfaces for L ^A T _E X	839
1.7	A sample package for structuring the log output	839
2	The Implementation	840
2.1	Document and package-level commands	840
2.2	<code>expl3</code> helpers	841
2.3	Declaring the file-related hooks	844
2.4	Patching L ^A T _E X's <code>\InputIfFileExists</code> command	844
2.5	Declaring a file substitution	846
2.6	Selecting a file (<code>\set@curr@file</code>)	848
2.7	Replacing a file and detecting loops	851
2.7.1	The Tortoise and Hare algorithm	852
2.8	Preventing a package from loading	853
2.9	High-level interfaces for L ^A T _E X	854
2.10	Internal commands needed elsewhere	854
3	A sample package for structuring the log output	855
4	Package emulations	856
4.1	Package <code>atveryend</code> emulation	856
U	ltshipout.dtx	858

1	Introduction	858
1.1	Overloading the <code>\shipout</code> primitive	858
1.2	Provided hooks	859
1.3	Legacy L<small>A</small>T<small>E</small>X commands	861
1.4	Special commands for use inside the hooks	861
1.5	Provided L<small>u</small>aT<small>e</small>X callbacks	862
1.6	Information counters	862
1.7	Debugging <code>shipout</code> code	863
2	Emulating commands from other packages	863
2.1	Emulating <code>atbegshi</code>	863
2.2	Emulating <code>everyshi</code>	864
2.3	Emulating <code>atenddvi</code>	864
2.4	Emulating <code>everypage</code>	865
3	The Implementation	865
3.1	Debugging	865
3.2	Handling the end of job hook	877
4	Legacy L<small>A</small>T<small>E</small>X 2ε interfaces	880
5	Internal commands needed elsewhere	881
6	Package emulation for compatibility	882
6.1	Package <code>atenddvi</code> emulation	882
6.2	Package <code>atbegshi</code> emulation	883
6.3	Package <code>everyshi</code> emulation	884
V	loutput.dtx	885
1	Output Routine	885
1.1	Floats	885
1.1.1	Kludgeins	940
1.1.2	Float control	943
1.1.3	Float placement parameters	955
W	lhyphen.dtx	959
X	ltfinal.dtx	961

1	Final settings	961
1.1	Debugging	961
1.2	Typesetting parameters	961
1.3	Lccodes for hyphenation	964
1.4	Hyphenation	967
1.5	Font loading	968
1.6	Input encoding	969
1.7	Lccodes and uccodes	973
1.8	Applying Patch files	975
1.9	Freeing Memory	976
1.10	Initialise file list	977
1.11	Preparation for supporting PDF in backends	977
1.12	Do some temporary work for pre-release	977
1.13	Some last minute initializations	977
1.14	Dumping the format	978
Change History		979
Index		1047

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 ≤ *v* ≤ 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.

2.1 INITEX

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

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.

```
8 \catcode`\}=2
```

If \LaTeX is in use the extensions and other new primitives have to be activated: this is done as early as possible. Older versions of \LaTeX 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!

```
9 \ifx\directlua\undefined
```

```
10 \else
```

```
11 \ifx\luatexversion\undefined
```

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

```
12 \directlua{tex.enableprimitives("",%
```

```
13 tex.extraprimitives('etex', 'pdftex', 'umath'))}
```

In current formats enable primitives with unprefixed names. the $\text{\textralatexrelease}$ guards allow the primitives to be defined with a \luatex prefix if older formats are specified.

```
14 </initex>
```

```
15 </dircheck>
```

```
16 <*initex, latexrelease>
```

```
17 <latexrelease>\ifx\directlua\undefined\else
```

```
18 <latexrelease>\IncludeInRelease{2015/10/01}{\luatexluafunction}
```

```
19 <latexrelease> {LuaTeX (prefixed names)}%
```

```
20 \directlua{tex.enableprimitives("",%
```

```
21 tex.extraprimitives("omega", "aleph", "luatex"))}
```

```
22 <latexrelease>\EndIncludeInRelease
```

```
23 <latexrelease>\IncludeInRelease{0000/00/00}{\luatexluafunction}
```

```
24 <latexrelease> {LuaTeX (prefixed names)}%
```

```
25 <latexrelease>\directlua{
```

```
26 <latexrelease> tex.enableprimitives(
```

```
27 <latexrelease> "luatex",
```

```
28 <latexrelease> tex.extraprimitives("core", "omega", "aleph", "luatex")
```

```
29 <latexrelease> )
```

```
30 <latexrelease> local i
```

```
31 <latexrelease> local t = { }
```

```
32 <latexrelease> for _,i in pairs(tex.extraprimitives("luatex")) do
```

```
33 <latexrelease> if not string.match(i,"^U") then
```

```
34 <latexrelease> if not string.match(i, "^luatex") then
```

```
35 <latexrelease> table.insert(t,i)
```

```
36 <latexrelease> end
```

```
37 <latexrelease> else
```

```
38 <latexrelease> if string.match(i,"^Uchar$") then
```

```
39 <latexrelease> table.insert(t,i)
```

```
40 <latexrelease> end
```

```
41 <latexrelease> end
```

```
42 <latexrelease> end
```

```
43 <latexrelease> for _,i in pairs(t) do
```

```
44 <latexrelease> tex.print(
```

```
45 <latexrelease> "\noexpand\\let\\noexpand\\\" .. i
```

```
46 <latexrelease> .. "\noexpand\\undefined"
```

```
47 <latexrelease> )
```

```
48 <latexrelease> end
```

```
49 <latexrelease> }
```

```
50 <latexrelease>\EndIncludeInRelease
```

```
51 <latexrelease>\fi
```

```
52 </initex, latexrelease>
```

```
53 <*dircheck>
```

```
54 <*initex>
```

```

55   \fi
56 \fi
      A test can now be made for eTeX.
57 <initex> \ifx\TeXversion\undefined
58 <initex> \errmessage
59 <initex> {LaTeX requires e-TeX}
60 <initex> \expandafter\endinput
61 <initex> \fi
      That distraction over, back to the basics of a format.
62 \catcode`\#=6
63 \catcode`\^=7
64 \chardef\active=13
65 \catcode`\@=11
66 \countdef\count@=255
67 \let\bgroup=\let\egroup=
68 \ifx\@input\@undefined\let\@input\input\fi
69 \ifx\@end\@undefined\let\@end\end\fi
70 \chardef\@inputcheck0
71 \chardef\sixt@n=16
72 \newlinechar`\^J
73 \def\typeout{\immediate\write17}
74 \def\dospecials{\do\ \do\\\do{\{do\}}\do\$\\do\&%
75 \do#\do\^\\do\_\\do%\do\~}
76 \def\@makeother#1{\catcode`#1=12\relax}
77 \def\space{ }
78 \def\@tempswafalse{\let\if@tempswa\iffalse}
79 \def\@tempswatrue{\let\if@tempswa\iftrue}
80 \let\if@tempswa\iffalse
81 \def\loop#1\repeat{\def\iterate{\relax\expandafter\iterate\fi}%
82 \iterate\let\iterate\relax}
83 \let\repeat\fi
84 </initex>

```

2.2 Some bits of 2e

```

85 {*2ekernel}
86 \def\two@digits#1{\ifnum#1<10 0\fi\number#1}
87 \long\def\@firstoftwo#1#2{#1}
88 \long\def\@secondoftwo#1#2{#2}

```

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

```

89 \def\ProvidesFile#1{%
90   \begingroup
91     \catcode`\ 10 %
92     \ifnum \endlinechar<256 %
93       \ifnum \endlinechar>\m@ne
94         \catcode\endlinechar 10 %
95       \fi
96     \fi
97     \@makeother\%
98     \@ifnextchar[{\@providesfile{#1}}{\@providesfile{#1}[]}
99   \def\@providesfile#1[#2]{%
100     \wlog{File: #1 #2}%
101     \@addtofilelist{ #2}%
102   \endgroup}

```

```

103 \long\def\@addtolist#1{%
104 \def\@empty{}%
105 \catcode`\%=12%
106 \def\@percentchar{\%}%
107 \catcode`\%=14%
108 \let\@currdir\@undefined%
109 \let\input@path\@undefined%
110 \let\filename@parse\@undefined%

```

\strip@prefix

```

111 \def\strip@prefix#1>{}%
112 </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.

```

113 <*docstrip>
114 \openin15=texsys.cfg
115 \ifeof15
116 \typeout{** Writing a default texsys.cfg}
117 \immediate\openout15=texsys.cfg
118 \begingroup
119 \catcode`^^M\active%
120 \let^^M\par%
121 \def\reserved@a#1^^M{%
122 \def\reserved@b{#1}%
123 \ifx\reserved@b\reserved@c\endgroup\else%
124 \immediate\write15{#1}%
125 \expandafter\reserved@a\fi}%
126 \def\reserved@d#1START^^M{\let\do\@makeother\dospecials\reserved@a}%
127 \catcode`\%=12%
128 \def\reserved@c{\%END}%
129 \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 $\langle\text{filename}\rangle\langle\text{space}\rangle$ 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 $\backslash\text{def}\backslash\text{@currdir}\{\cdot/\}$. For more exotic operating systems you may want to make \@currdir a macro with arguments delimited by . and/or $\langle\text{space}\rangle$. 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 $\langle\text{dir}\rangle$ is an entry in the input path, TeX will try to load the expansion of $\langle\text{dir}\rangle\langle\text{filename}\rangle\langle\text{space}\rangle$

So either $\langle\text{dir}\rangle$ 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 $\langle\text{filename}\rangle$. This means that for UNIX-like syntax, each $\langle\text{dir}\rangle$ 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**. L^AT_EX 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.

```
130 %\def\@currdir{\cdot/}
131 %\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 `/`.

```
132 % \def\@currdir{./}
133 % \def\input@path{%
134 %   {/usr/local/lib/tex/inputs/distrib/}%
135 %   {/usr/local/lib/tex/inputs/contrib/}%
136 %   {/usr/local/lib/tex/inputs/local/}%
137 % }
```

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.

```
138 % \def\@currdir{./}
139 % \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.

```
140 % \def\@currdir{./}
141 % \def\input@path{%
142 %   {c:/tex/inputs/distrib/}%
143 %   {c:/tex/inputs/contrib/}%
144 %   {c:/tex/inputs/local/}%
145 % }
```

3.6 VMS (DECUS TEX, 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.

```
146 % \def\@currdir{[]}
147 % \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:

```
148 % \def\@currdir{[]}
149 % \def\input@path{%
150 %   {tex_inputs:}%
151 %   {SOMEDISK:[SOME.TEX.DIRECTORY]}%
152 % }
```

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.

```
153 % \def\@currdir{:  
154 % \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.

```
155 % \def\@currdir{:  
156 % \def\input@path{  
157 %   {Hard-Disk:Applications:TeX:TeX-inputs:  
158 %   {Hard-Disk:Applications:TeX:My-inputs:  
159 % }
```

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:

```
160 % \def\@dir#1#2 {  
161 %   \@d@r{#1}#2..\@nil}  
162 % \def\@d@r#1#2.#3.#4@\nil{  
163 %   <\ifx\@dir#1\@dir\else#1\ifx\@dir#3\@dir\else.\fi\fi#3>#2 }  
164 %  
165 % \def\@currdir{\@dir{}}  
166 % \def\input@path{  
167 %   {\@dir{area.one}}%  
168 %   {\@dir{area.two}}%  
169 % }
```

END

```
170 \immediate\closeout15
```

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

```
171 \else  
172 \typeout{** Using the existing texsys.cfg}  
173 \closein15  
174 \input texsys.cfg  
175 \fi  
176 </docstrip>
```

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

```
177 <dircheck>\input texsys.cfg
```

4 Setting \@currdir

\@currdir
\IfFileExists This is a local definition of \IfFileExists. It tries to relocate `texssys.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.

```
178  \begingroup
179  \count@ \time
180  \divide \count@ 60
181  \count2=-\count@
182  \multiply \count2 60
183  \advance \count2 \time
```

The current date and time stamp.

```
184  \edef \today{%
185    \the \year / \two@digits{\the \month} / \two@digits{\the \day} : %
186    \two@digits{\the \count@} : \two@digits{\the \count2}}
```

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

```
187  \immediate \openout 15 = texsys.aux
188  \immediate \write 15 {\today^J}
189  \immediate \closeout 15 %
```

#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!

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

If \@currdir has not been pre-defined in `texsys.cfg` then test for UNIX, VMS and Oz-T_EX-Mac. syntax.

```
205  \ifx \a @currdir \undefined
206    \IfFileExists {./texsys.aux} {\gdef \a @currdir {./}} %
207    {\IfFileExists {} {texsys.aux} {\gdef \a @currdir {}}} %
208    {\IfFileExists {::texsys.aux} {\gdef \a @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.

```

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

```

Otherwise `\@currdir` was defined in `texsys.cfg`. In this case check that the syntax specified works on this system. (In case a complete L^AT_EX 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.

```

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

```

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

```

223 \fi
224 \immediate\closeout15 %
225 \endgroup
226 \typeout{^^J^^J%
227   \noexpand\@currdir set to:
228   \expandafter\strip@prefix\meaning\@currdir.^^J%
229 }

```

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

Stop here if the file is being used unstripped.

```

230 <*docstrip>
231 \relax\endinput
232 </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.

```

233 \typeout{^^J%

```

```

234     Assuming \noexpand\openin and \noexpand\input^^J%
235     \ifx\input@path\@undefined
236         have the same search path.^^J%
237     \else
238         \input@path has been defined in texsys.cfg.
239             have different search paths.^^J%
240             LaTeX will use the path specified by \noexpand\input@path:^^J%
241         \fi
242     \}
243
244 (End definition for \input@path.)
```

6 Filename Parsing

\filename@parse Split a filename into its components.

```

242 \ifx\filename@parse\@undefined
243   \def\reserved@a{./}\ifx\@currdir\reserved@a
244 \filename@parse was not specified in texsys.cfg, but \@currdir looks like UNIX...
245   \typeout{^^JDefining UNIX/DOS style filename parser.^^J}
246   \def\filename@parse#1{%
247     \let\filename@area\empty
248     \expandafter\filename@path#1/\\"}
249
250 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
260 \else\def\reserved@a{}{}\ifx\@currdir\reserved@a
261 \filename@parse was not specified in texsys.cfg, but \@currdir looks like VMS...
262   \typeout{^^JDefining VMS style filename parser.^^J}
263   \def\filename@parse#1{%
264     \let\filename@area\empty
265     \expandafter\filename@path#1\\"}
266
267 Search for the last ].
268   \def\filename@path#1]#2\\{%
269     \ifx\\#2\\%
270       \def\reserved@a{\filename@simple#1.\\"}%
271     \else
272       \edef\filename@area{\filename@area#1}%
273       \def\reserved@a{\filename@path#2\\"}%
274     \fi
275   \reserved@a}
276
277 \else\def\reserved@a{}{}\ifx\@currdir\reserved@a
```

```

\filename@parse was not specified in texsys.cfg, but \currdir looks like Macintosh...
270      \typeout{^^JDefining Mac style filename parser.^^J}
271      \def\filename@parse#1{%
272          \let\filename@area\empty
273          \expandafter\filename@path#1:\ //}
274      Search for the last :.
275      \def\filename@path#1:#2\\{%
276          \ifx\\#2\\%
277              \def\reserved@a{\filename@simple#1.\ //}
278          \else
279              \edef\filename@area{\filename@area#1:}%
280              \def\reserved@a{\filename@path#2\\}%
281          \fi
282          \reserved@a}
283      \else
284      \typeout{^^JDefining generic filename parser.^^J}
285      \def\filename@parse#1{%
286          \let\filename@area\empty
287          \expandafter\filename@simple#1.\ //}
288      \fi\fi\fi
289      \filename@simple is used by all three versions. Finally we can split off the extension.
290      </dircheck>
291      <*dircheck, latexrelease>
292      <(latexrelease)>\IncludeInRelease{2019/10/01}{\filename@simple}
293          {Final dot for extension}%
294      \def\filename@simple#1.#2\\{%
295          \ifx\\#2\\%
296              \let\filename@ext\relax
297              \edef\filename@base{#1}%
298          \else
299              \filename@dots{#1}#2\\%
300          \fi}
301      \def\filename@dots#1.#2.#3\\{%
302          \ifx\\#3\\%
303              \def\filename@ext{#2}%
304              \edef\filename@base{#1}%
305          \else
306              \filename@dots{#1.#2}#3\\%
307          \fi}
308      <(latexrelease)>\EndIncludeInRelease
309      <(latexrelease)>\IncludeInRelease{0000/00/00}{\filename@simple}
310          {Final dot for extension}%
311      <(latexrelease)>    \def\filename@simple#1.#2\\{%
312          \ifx\\#2\\%
313              \let\filename@ext\relax
314              \else
315                  \edef\filename@ext{\filename@dot#2\\}%

```

```

314 <{latexrelease}>      \fi
315 <{latexrelease}>      \edef\filename@base{\#1}}
316 <{latexrelease}> \EndIncludeInRelease
317 </dircheck, latexrelease>
318 <{*dircheck}>

        Remove a final dot, added earlier.

319 \def\filename@dot#1.\{\#1}

320 \else

Otherwise, \filename@parse was specified in texsys.cfg.
321 \typeout{^^J^^J%
322   \noexpand\filename@parse was defined in texsys.cfg:^^J%
323   \expandafter\strip@prefix\meaning\filename@parse.^^J%
324 }
325 \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.

```

326 \ifx\@TeXversion\undefined
327   \ifx\@undefined\inputlineno
328     \def\@TeXversion{2}
329   \else
330     {\catcode`\\=active
331       \def\reserved@a{\if#1\string^3\fi}
332       \edef\reserved@a{\expandafter\reserved@a\string^J\@C}
333       \ifx\reserved@a\empty\else\gdef\@TeXversion{3}\fi}
334   \fi
335 \fi

(End definition for \@TeXversion.)
336 </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\float@count}  
52 \def\newdimen {\e@alloc\dimen \dimendef {\count11}\insc@unt\float@count}  
53 \def\newskip {\e@alloc\skip \skipdef {\count12}\insc@unt\float@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\float@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)}%}
              \gdef\c@ch@ck#1#2#3#4{%
156    \ifnum#1<#2\else
157      \fi

```

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 \c@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  {latexrelease\EndIncludeInRelease
182  {latexrelease\IncludeInRelease{0000/00/00}%
183  {latexrelease}{\e@ch@ck}{Extended Allocation (checking)}%
184  {latexrelease\let\c@ch@ck\c@undefined
185  {latexrelease\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@>\c@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 </2ekernel>
 346 <*2ekernel | latexrelease>
 347 <latexrelease>\IncludeInRelease{2021/06/01}{\tracingstacklevels}%
 348 <latexrelease>          {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 <latexrelease>\EndIncludeInRelease
 356 <latexrelease>
 357 <latexrelease>\IncludeInRelease{0000/00/00}{\tracingstacklevels}%
 358 <latexrelease>          {tracingstacklevels}%
 359 <latexrelease>\ifx\directlua\@undefined
 360 <latexrelease>\else
 361 <latexrelease>  \let\tracingstacklevels\@undefined
 362 <latexrelease>\fi
 363 <latexrelease>\EndIncludeInRelease
 364 </2ekernel | latexrelease>
 365 <*2ekernel>

\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 391 \newskip\normalbaselineskip \normalbaselineskip=12pt
  \normallineskip 392 \newskip\normallineskip \normallineskip=1pt
  \normallineskiplimit 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 \@m\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 \lineskip\limits\normallineskip\limits}`

(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
```

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

(End definition for `\endgraf` and `\endline`. These functions are documented on page 292.)

```
\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=\par` instead of `\def^M{\par}` since this allows, for example, `\let\par=\cr \obeylines \halign{...}`

```
419 {\catcode`^M=\active % these lines must end with %  
420 \gdef\obeylines{\catcode`^M\active \let^M\par} %  
421 \global\let^M\par} % this is in case ^M appears in a \write  
422 \def\obeyspaces{\catcode`\ \active}  
423 {\obeyspaces\global\let =\space}
```

(End definition for `\obeylines` and `\obeyspaces`.)

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

```
424 \long\def \loop #1\repeat{  
425   \def\iterate{\#1\relax % Extra \relax  
426             \expandafter\iterate\fi  
427           }%  
428   \iterate  
429   \let\iterate\relax  
430 }
```

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

```
431 \let\repeat=\fi
```

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

L^AT_EX defines `\smallskip`, etc. in `ltxspace.dtx`.

```

\nointerlineskip
\offinterlineskip 432 \def\nointerlineskip{\prevdepth-\@m\p@}
433 \def\offinterlineskip{\baselineskip-\@m\p@}
434 \lineskip\z@\lineskiplimit\maxdimen}

(End definition for \nointerlineskip and \offinterlineskip.)

\vglue
\hglue 435 \def\vglue{\afterassignment\vgl@{\skip@=}}
436 \def\vgl@{\par \dimen@\prevdepth \hrule \height\z@
437 \nobreak\vskip\skip@ \prevdepth\dimen@}
438 \def\hglue{\afterassignment\hgl@{\skip@=}}
439 \def\hgl@{\leavevemode \count@\spacefactor \vrule \width\z@
440 \nobreak\hskip\skip@ \spacefactor\count@}

(End definition for \vglue and \hglue.)
 $\text{\LaTeX}$  defines ~ in ltdefns.dtx.

\slash This generates a / acting a bit like - but still allows hyphenation in the word part preceding it (but not after).
441 \def\slash{/penalty\exhyphenpenalty}

(End definition for \slash.)

\break
\nobreak 442 \def\break{\penalty-\@M}
\allowbreak 443 \def\nobreak{\penalty \@M}
444 \def\allowbreak{\penalty \z@}

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

\filbreak
\goodbreak 445 \def\filbreak{\par\vfil\penalty-200\vfilneg}
446 \def\goodbreak{\par\penalty-500 }

(End definition for \filbreak and \goodbreak.)

\eject Define \eject as in plain  $\text{\TeX}$  but define \supereject only in the compatibility file.
447 \def\eject{\par\break}

(End definition for \eject.)

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

(End definition for \removelastskip.)

\smallbreak
\medbreak 449 \def\smallbreak{\par\ifdim\lastskip<\smallskipamount
450 \removelastskip\penalty-50\smallskip\fi}
\bigbreak 451 \def\medbreak{\par\ifdim\lastskip<\medskipamount
452 \removelastskip\penalty-100\medskip\fi}
453 \def\bigbreak{\par\ifdim\lastskip<\bigskipamount
454 \removelastskip\penalty-200\bigskip\fi}

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

```

```

\m@th
455 \def\m@th{\mathsurround\z@}

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

\underline Due to L^AT_EX's redefinition of \underline plain T_EX's \underline can be done in a simpler fashion (but do we need it at all?).

```

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

(End definition for \underline.)

\strutbox L^AT_EX sets \strutbox in \set@fontsize.

```

\strut
457 \newbox\strutbox
458 \def\strut{\relax\ifmmode\copy\strutbox\else\unhcopy\strutbox\fi}
```

(End definition for \strutbox and \strut.)

\hidewidth For alignment entries that can stick out.

```

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

(End definition for \hidewidth.)

\narrower

```

460 \def\narrower{%
461   \advance\leftskip\parindent
462   \advance\rightskip\parindent}
```

(End definition for \narrower.)

L^AT_EX defines \ae and similar commands elsewhere.

```

463 \chardef\%='\%
464 \chardef\&='\&
465 \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

```

466 \def\leavevmode{\unhbox\vvoidbox}
```

(End definition for \leavevmode.)

\mathhexbox

```

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

(End definition for \mathhexbox.)

\ialign

```

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

(End definition for \ialign.)

\oalign

```

\o@align
469 \def\oalign#1{\leavevmode\vtop{\baselineskip\z@skip \lineskip.25ex%
470   \ialign{\##\crcr#1\crcr}}}
471 \def\o@align{\lineskip\z@ \oalign}
472 \def\ooalign{\lineskip\z@-\maxdimen \oalign}
```

(End definition for `\oalign`, `\o@ign`, 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 L^AT_EX.

Since the version given here has been in use by L^AT_EX 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.

```
473 \def\sh@ft#1{\dimen@.00#1ex\multiply\dimen@\fontdimen1\font  
474   \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).

```
475 \def\ltx@sh@ft #1{  
476   \dimen@ #1%  
477   \kern \strip@pt  
478   \fontdimen1\font \dimen@  
479 } % 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.

```
480 \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.

```
481 \def\dotfill{  
482   \leavevmode  
483   \cleaders \hb@xt@ .44em{\hss.\hss}\hfill  
484   \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:

```
485 \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:
486 \def\showoverfull{\tracingonline@ne}

(End definition for \showoverfull.)

\showoutput
\loggingoutput
487 \gdef\loggingoutput{\tracingoutput@ne
488   \showboxbreadth\maxdimen\showboxdepth\maxdimen\errorstopmode}
489 \gdef\showoutput{\loggingoutput\showoverfull}
490 (/2ekernel)

(End definition for \showoutput and \loggingoutput.)

\tracingall
\loggingall
491 <latexrelease>\IncludeInRelease{2021/06/01}{\loggingall}
492 <latexrelease>          {\tracingstacklevels and \tracinglostchars=3}%
493 <*2ekernel | latexrelease>
494 \edef\loggingall{%
495   \tracingstats\tw@
496   \tracingpages@ne
497   \tracinglostchars\thr@@
498   \tracingparagraphs@ne
499   \tracinggroups@ne
500   \tracingifs@ne
501   \tracingscantokens@ne
502   \tracingnesting@ne
503   \errorcontextlines\maxdimen
504   \ifdefinable\tracingstacklevels {\tracingstacklevels\maxdimen \fi
505   \noexpand \loggingoutput
506   \tracingmacros\tw@
507   \tracingcommands\thr@@
508   \tracingrestores@ne
509   \tracingassigns@ne
510 }%
511 \def\tracingall{\showoverfull\loggingall}
512 (/2ekernel | latexrelease)
513 <latexrelease>\EndIncludeInRelease
514 <latexrelease>
515 <latexrelease>\IncludeInRelease{2015/01/01}{\loggingall}{etex tracing}%
516 <latexrelease>\ifx\tracingscantokens@undefined
517 <latexrelease>\gdef\loggingall{%
518 <latexrelease> \tracingstats\tw@
519 <latexrelease> \tracingpages@ne
520 <latexrelease> \tracinglostchars@ne
521 <latexrelease> \tracingparagraphs@ne
522 <latexrelease> \errorcontextlines\maxdimen
523 <latexrelease> \loggingoutput
524 <latexrelease> \tracingmacros\tw@
525 <latexrelease> \tracingcommands\tw@
526 <latexrelease> \tracingrestores@ne
527 <latexrelease>}%
528 <latexrelease>\else
529 <latexrelease>\gdef\loggingall{%
530 <latexrelease> \tracingstats\tw@

```

```

531 〈\latexrelease〉 \tracingpages\@ne
532 〈\latexrelease〉 \tracinglostchars\tw@
533 〈\latexrelease〉 \tracingparagraphs\@ne
534 〈\latexrelease〉 \tracinggroups\@ne
535 〈\latexrelease〉 \tracingifs\@ne
536 〈\latexrelease〉 \tracingscantokens\@ne
537 〈\latexrelease〉 \tracingnesting\@ne
538 〈\latexrelease〉 \errorcontextlines\maxdimen
539 〈\latexrelease〉 \loggingoutput
540 〈\latexrelease〉 \tracingmacros\tw@
541 〈\latexrelease〉 \tracingcommands\thr@@
542 〈\latexrelease〉 \tracingrestores\@ne
543 〈\latexrelease〉 \tracingassigns\@ne
544 〈\latexrelease〉}%
545 〈\latexrelease〉\fi
546 〈\latexrelease〉\gdef\tracingall{\showoverfull\loggingall}
547 〈\latexrelease〉\EndIncludeInRelease
548 〈\latexrelease〉
549 〈\latexrelease〉\IncludeInRelease{0000/00/00}{\loggingall}{etex tracing}%
550 〈\latexrelease〉\gdef\loggingall{\tracingcommands\tw@\tracingstats\tw@
551 〈\latexrelease〉 \tracingpages\@ne\tracinglostchars\@ne
552 〈\latexrelease〉 \tracingmacros\tw@\tracingparagraphs\@ne\tracingrestores\@ne
553 〈\latexrelease〉 \errorcontextlines\maxdimen\loggingoutput}
554 〈\latexrelease〉 \gdef\tracingall{\loggingall\showoverfull}
555 〈\latexrelease〉\EndIncludeInRelease

```

(*End definition for \tracingall and \loggingall.*)

```
\tracingnone
556 〈\latexrelease〉\IncludeInRelease{2015/01/01}{\tracingnone}%
557 〈\latexrelease〉                                         {turn off etex tracing}%
558 〈*2ekernel | \latexrelease〉
559 \edef\tracingnone{%
560   \tracingassigns\z@%
561   \tracingrestores\z@%
562   \tracingonline\z@%
563   \tracingcommands\z@%
564   \showboxdepth\m@ne%
565   \showboxbreadth\m@ne%
566   \tracingoutput\z@%
567   \errorcontextlines\m@ne%
568   \ifdefined\tracingstacklevels \tracingstacklevels\z@ \fi%
569   \tracingnesting\z@%
570   \tracingscantokens\z@%
571   \tracingifs\z@%
572   \tracinggroups\z@%
573   \tracingparagraphs\z@%
574   \tracingmacros\z@%
575   \tracinglostchars\@ne%
576   \tracingpages\z@%
577   \tracingstats\z@%
578 }%
579 〈/2ekernel | \latexrelease〉
580 〈\latexrelease〉\EndIncludeInRelease

```

```

581 〈\latexrelease〉
582 〈\latexrelease〉\IncludeInRelease{2015/01/01}{\tracingnone}%
583 〈\latexrelease〉                                              {turn off etex tracing}%
584 〈\latexrelease〉\ifx\tracingscantokens\@undefined
585 〈\latexrelease〉\def\tracingnone{%
586 〈\latexrelease〉  \tracingonline\z@%
587 〈\latexrelease〉  \tracingcommands\z@%
588 〈\latexrelease〉  \showboxdepth\m@ne%
589 〈\latexrelease〉  \showboxbreadth\m@ne%
590 〈\latexrelease〉  \tracingoutput\z@%
591 〈\latexrelease〉  \errorcontextlines\m@ne%
592 〈\latexrelease〉  \tracingrestores\z@%
593 〈\latexrelease〉  \tracingparagraphs\z@%
594 〈\latexrelease〉  \tracingmacros\z@%
595 〈\latexrelease〉  \tracinglostchars\@ne%
596 〈\latexrelease〉  \tracingpages\z@%
597 〈\latexrelease〉  \tracingstats\z@%
598 〈\latexrelease〉}%
599 〈\latexrelease〉\else
600 〈\latexrelease〉\def\tracingnone{%
601 〈\latexrelease〉  \tracingassigns\z@%
602 〈\latexrelease〉  \tracingrestores\z@%
603 〈\latexrelease〉  \tracingonline\z@%
604 〈\latexrelease〉  \tracingcommands\z@%
605 〈\latexrelease〉  \showboxdepth\m@ne%
606 〈\latexrelease〉  \showboxbreadth\m@ne%
607 〈\latexrelease〉  \tracingoutput\z@%
608 〈\latexrelease〉  \errorcontextlines\m@ne%
609 〈\latexrelease〉  \tracingnesting\z@%
610 〈\latexrelease〉  \tracingscantokens\z@%
611 〈\latexrelease〉  \tracingifs\z@%
612 〈\latexrelease〉  \tracinggroups\z@%
613 〈\latexrelease〉  \tracingparagraphs\z@%
614 〈\latexrelease〉  \tracingmacros\z@%
615 〈\latexrelease〉  \tracinglostchars\@ne%
616 〈\latexrelease〉  \tracingpages\z@%
617 〈\latexrelease〉  \tracingstats\z@%
618 〈\latexrelease〉}%
619 〈\latexrelease〉\fi
620 〈\latexrelease〉\EndIncludeInRelease
621 〈\latexrelease〉
622 〈\latexrelease〉\IncludeInRelease{0000/00/00}{\tracingnone}%
623 〈\latexrelease〉                                              {turn off etex tracing}%
624 〈\latexrelease〉\let\tracingnone\@undefined
625 〈\latexrelease〉\EndIncludeInRelease

```

(End definition for \tracingnone.)

\hideoutput

```

626 〈*2ekernel | \latexrelease〉
627 〈\latexrelease〉\IncludeInRelease{2015/01/01}{\hideoutput}%
628 〈\latexrelease〉                                              {hide output from tracing}%
629 〈\def\hideoutput{%
630    \tracingoutput\z@%

```

```

631   \showboxbreadth\m@ne
632   \showboxdepth\m@ne
633   \tracingonline\m@ne
634 }%
635 <| latexrelease |> \EndIncludeInRelease
636 <| latexrelease |>
637 <| latexrelease |> \IncludeInRelease{0000/00/00}{\hideoutput}%
638 <| latexrelease |>                                         {hide output from tracing}%
639 <| latexrelease |> \let\hideoutput\@undefined
640 <| latexrelease |> \EndIncludeInRelease
641 </2ekernel | latexrelease>

(End definition for \hideoutput.)
ITEX change: \showhyphens Defined later.
Punctuation affects the spacing.

642 <*2ekernel>
643 \nonfrenchspacing
644 </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  \langle latexrelease \rangle \edef\latexreleaseversion
6  {*2ekernel | latexrelease}
7  {2021-11-15}
8  \langle /2ekernel | latexrelease \rangle
9  \langle *2ekernel \rangle
10 \def\patch@level{1}
```

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.

```
\development@branch@name
11 \edef\development@branch@name{}
```

(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  \langle latexrelease \rangle \newif\if@includeinrelease
71  \langle latexrelease \rangle \@includeinreleasefalse
72  \def\IncludeInRelease#1{%
73    \if@includeinrelease
74      \PackageError{latexrelease}{mis-matched \IncludeInRelease}%
75      {There is an \string\EndIncludeRelease\space missing}%
76    \elsefalse
77    \fi
78    \ifnum0%
79      \ifx\new@moduledate\empty\else 1\fi
80      \ifnum \expandafter\@parse@version#1//00\@nil=0 1\fi
81      =11
82      \expandafter\@firstoftwo
83    \else
84      \expandafter\@secondoftwo
85    \fi
86    {\@finish@module@release{#1}}%
87    {\@kernel@ifnextchar[%
88      {\@IncludeInRelease{#1}}
89      {\@IncludeInRelease{#1}[#1]}}}
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}
110
If a specific date has not been specified in latexrelease use '#1'.
111 \def\@IncludeInRelease#1[#2]{\@IncludeInRelease{#2}}
112 \def\@IncludeInRelease#1#2#3{%
113   \toks@{[#1] #3}%
114   \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{\IncludeInRelease}
178 \def\EndModuleRelease{%
179     \ifx\new@modulename\@empty
180         \@latex@error{Extra \string\EndModuleRelease .}\@eha
181     \else
182         \GenericInfo{}{END module: \new@modulename\space (\new@moduledate)}%
183         \let\new@modulename\@empty
184         \let\new@moduledate\@empty
185         \@skipping@modulefalse
186     \fi}
187 
```

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

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).

\newwhatsit \newwhatsit{\langle whatsit\rangle}
Defines a custom \whatsit, indexed from 1.

\newluabytecode \newluabytecode{\langle bytecode\rangle}
```

<code>\newluachunkname</code>	Allocates a number for Lua bytecode register, indexed from 1. <code>newluachunkname{\langle chunkname\rangle}</code>
<code>\catcodetable@initex</code>	Allocates a number for Lua chunk register, indexed from 1. Also enters the name of the register (without backslash) into the <code>lua.name</code> table to be used in stack traces.
<code>\catcodetable@string</code>	Predefined category code tables with the obvious assignments. Note that the <code>latex</code> and <code>atletter</code> tables set the full Unicode range to the codes predefined by the kernel.
<code>\catcodetable@latex</code>	<code>\setattribute{\langle attribute\rangle}{\langle value\rangle}</code>
<code>\catcodetable@atletter</code>	<code>\unsetattribute{\langle attribute\rangle}</code>
<code>\setattribute</code>	Set and unset attributes in a manner analogous to <code>\setlength</code> . Note that attributes take a marker value when unset so this operation is distinct from setting the value to zero.
<code>\unsetattribute</code>	

3 Plain T_EX interface

The `ltluatex` interface may be used with plain T_EX using `\input{ltluatex}`. This inputs `ltluatex.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 `ltluatex` 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 `ltluatex` code but implements a compatibility layer providing the interface of the original package.

4 Lua functionality

4.1 Allocators in Lua

<code>new_attribute</code>	<code>luatexbase.new_attribute{\langle attribute\rangle}</code>
	Returns an allocation number for the <code>\langle attribute\rangle</code> , indexed from 1. The attribute will be initialised with the marker value <code>-"7FFFFFFF"</code> ('unset'). The attribute allocation sequence is shared with the T _E X code but this function does <i>not</i> define a token using <code>\attributedef</code> . The attribute name is recorded in the <code>attributes</code> table. A metatable is provided so that the table syntax can be used consistently for attributes declared in T _E X or Lua.
<code>new_whatsit</code>	<code>luatexbase.new_whatsit{\langle whatsit\rangle}</code>
	Returns an allocation number for the custom <code>\langle whatsit\rangle</code> , indexed from 1.
<code>new_bytecode</code>	<code>luatexbase.new_bytecode{\langle bytecode\rangle}</code>
	Returns an allocation number for a bytecode register, indexed from 1. The optional <code>\langle name\rangle</code> argument is just used for logging.
<code>new_chunkname</code>	<code>luatexbase.new_chunkname{\langle chunkname\rangle}</code>
	Returns an allocation number for a Lua chunk name for use with <code>\directlua</code> and <code>\latelua</code> , indexed from 1. The number is returned and also <code>\langle name\rangle</code> argument is added to the <code>lua.name</code> array at that index.
<code>new_luafunction</code>	<code>luatexbase.new_luafunction{\langle functionname\rangle}</code>
	Returns an allocation number for a lua function for use with <code>\luafunction</code> , <code>\lateluafunction</code> , and <code>\luadef</code> , indexed from 1. The optional <code>\langle functionname\rangle</code> argument is just used for logging.

These functions all require access to a named T_EX count register to manage their allocations. The standard names are those defined above for access from T_EX,

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.registernumber(<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 `\TeX`'s internal tables. After for example `\newattribute\myattrib`, `\myattrib` would be defined by (say) `\myattrib=\attribute15`. `luatexbase.registernumber("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.registernumber("#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\TeX` 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 `<description>` from the `<callback>`. The removed function and its description are returned as the results of this function.

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

<code>disable_callback</code>	<code>luatexbase.disable_callback(<callback>)</code> Sets the <code><callback></code> to <code>false</code> as described in the LuaTeX manual for the underlying <code>callback.register</code> built-in. Callbacks will only be set to false (and thus be skipped entirely) if there are no functions registered using the callback.
<code>callback_descriptions</code>	A list of the descriptions of functions registered to the specified callback is returned. <code>{}</code> is returned if there are no functions registered.
<code>create_callback</code>	<code>luatexbase.create_callback(<name>,metatype,<default>)</code> Defines a user defined callback. The last argument is a default function or <code>false</code> .
<code>call_callback</code>	<code>luatexbase.call_callback(<name>,...)</code> 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 LuaTeX version

LuaTeX 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, LuaTeX will correctly find Lua files in the `texmf` tree without ‘help’.

```

3  \ifx\directlua\@undefined\else
4    \newluafunction{LuaTeX}%
5    \ifnum\luatexversion<60 %
6      \wlog{*****}
7      \wlog{* LuaTeX version too old for ltluatex support *}
8      \wlog{*****}
9      \expandafter\endinput
10 \fi

```

Two simple L^AT_EX 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 L^AT_EX/Plain T_EX setup

```

13  {*tex}

```

Older L^AT_EX 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 L^AT_EX, or plain T_EX, 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/LaTeX differences in `ltluatex.`)

```

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\@alloc@top=65535
48 \let\@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\@cclvi
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 LATEX.

72 \fi
73 </tex>

```

5.3 Attributes

`\newattribute` As is generally the case for the LuaT_EX 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 LuaT_EX 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   \@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 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\setstrangeccatcode#1#2#3{%
97     \ifnum#1>#2 %
98       \expandafter\@gobble
99     \else
100       \expandafter\@firstofone
101     \fi
102   {%
103     \catcode#1=#3 %
104     \expandafter\setstrangeccatcode\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   \setstrangeccatcode{65}{90}{12}%
114   \setstrangeccatcode{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@count\@undefined
174      \countdef\@alloc@luafunction@count=260
175      \@alloc@luafunction@count=\z@
176  \fi
177  \def\newluafunction{%
178      \@alloc@luafunction\@alloc@chardef
179      \@alloc@luafunction@count\m@ne\@alloc@top
180  }

```

(End definition for `\newluafunction`.)

5.6 Custom whatsits

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

```
181 \ifx\@alloc@whatsit@count\@undefined
182   \countdef\@alloc@whatsit@count=261
183   \@alloc@whatsit@count=\z@
184 \fi
185 \def\newwhatsit#1{%
186   \@alloc@whatsit\@alloc@chardef
187   \@alloc@whatsit@count\m@ne\@alloc@top#1%
188 }
```

(End definition for \newwhatsit.)

5.7 Lua bytecode registers

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

```
189 \ifx\@alloc@bytecode@count\@undefined
190   \countdef\@alloc@bytecode@count=262
191   \@alloc@bytecode@count=\z@
192 \fi
193 \def\newluabytecode#1{%
194   \@alloc@luabytecode\@alloc@chardef
195   \@alloc@bytecode@count\m@ne\@alloc@top#1%
196 }
```

(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.

```
197 \ifx\@alloc@luachunk@count\@undefined
198   \countdef\@alloc@luachunk@count=263
199   \@alloc@luachunk@count=\z@
200 \fi
201 \def\newluachunkname#1{%
202   \@alloc@luachunk\@alloc@chardef
203   \@alloc@luachunk@count\m@ne\@alloc@top#1%
204   {\escapechar\m@ne
205     \directlua{\lua.name[\the\allocationnumber]="\string#1"}%
206 }
```

(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:

```
207 \def\now@and@everyjob#1{%
208   \everyjob\expandafter{\the\everyjob
209     #1%
```

```

210 }%
211 #1%
212 }

```

Load the Lua code at the start of every job. For the conversion of TeX 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.

```

213 <2ekernel> \now@and@everyjob{%
214   \begingroup
215   \attributedef\attributezero=0 %
216   \chardef\charzero=0 %
217   \countdef\CountZero=0 %
218   \dimendef\dimenzero=0 %
219   \mathchardef\mathcharzero=0 %
220   \muskipdef\muskipzero=0 %
221   \skipdef\skipzero=0 %
222   \toksdef\tokszero=0 %
223   \directlua{require("ltluatex")}
224   \endgroup
225 <2ekernel> }
226 <|latexrelease>\EndIncludeInRelease

```

```

227 <|latexrelease>\IncludeInRelease{0000/00/00}
228 <|latexrelease>          {\newluafunction}{LuaTeX}%
229 <|latexrelease>\let\e@alloc@attribute@count\@undefined
230 <|latexrelease>\let\newattribute\@undefined
231 <|latexrelease>\let\setattribute\@undefined
232 <|latexrelease>\let\unsetattribute\@undefined
233 <|latexrelease>\let\e@alloc@ccodetable@count\@undefined
234 <|latexrelease>\let\newcatcodetable\@undefined
235 <|latexrelease>\let\catcodetable@initex\@undefined
236 <|latexrelease>\let\catcodetable@string\@undefined
237 <|latexrelease>\let\catcodetable@latex\@undefined
238 <|latexrelease>\let\catcodetable@atletter\@undefined
239 <|latexrelease>\let\e@alloc@luafunction@count\@undefined
240 <|latexrelease>\let\newluafunction\@undefined
241 <|latexrelease>\let\e@alloc@luafunction@count\@undefined
242 <|latexrelease>\let\newwhatsit\@undefined
243 <|latexrelease>\let\e@alloc@whatsit@count\@undefined
244 <|latexrelease>\let\newluabytecode\@undefined
245 <|latexrelease>\let\e@alloc@bytecode@count\@undefined
246 <|latexrelease>\let\newluachunkname\@undefined
247 <|latexrelease>\let\e@alloc@luachunk@count\@undefined
248 <|latexrelease>\directlua{luatexbase.uninstall()}%
249 <|latexrelease>\EndIncludeInRelease

```

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

```

250 <|latexrelease>\IncludeInRelease{2017/01/01}%
251 <|latexrelease>          {\fontencoding}{TU in everyjob}%
252 <|latexrelease>\fontencoding{TU}\let\encodingdefault\f@encoding
253 <|latexrelease>\ifx\directlua\@undefined\else
254 <2ekernel>\everyjob\expandafter{%
255 <2ekernel> \the\everyjob

```

```

256  {*2ekernel,latexrelease}
257  \directlua{%
258  if xpcall(function ()%
259      require('luaotfload-main')%
260      end,texio.write_nl) then %
261  local _void = luaotfload.main ()%
262  else %
263  texio.write_nl('Error in luaotfload: reverting to OT1')%
264  tex.print('\string\\def\string\\encodingdefault{OT1}')%
265  end %
266  }%
267  \let\f@encoding\encodingdefault
268  \expandafter\let\csname ver@luaotfload.sty\endcsname\fmtversion
269  //2ekernel,latexrelease)
270  \if latexrelease \fi
271  {2ekernel} %
272  \if latexrelease \EndIncludeInRelease
273  \if latexrelease \IncludeInRelease{0000/00/00}%
274  \if latexrelease {\fontencoding}{TU in everyjob}%
275  \if latexrelease \fontencoding{OT1}\let\encodingdefault\f@encoding
276  \if latexrelease \EndIncludeInRelease
277  {2ekernel | latexrelease} \fi
278  //2ekernel | tex | latexrelease)

```

5.10 Lua module preliminaries

279 `(*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.

```

280 luatexbase      = luatexbase or { }
281 local luatexbase = luatexbase

```

(*End definition for luatexbase.*)

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

```

282 local string_gsub      = string.gsub
283 local tex_count         = tex.count
284 local tex_setattribute = tex.setattribute
285 local tex_setcount       = tex.setcount
286 local texio_write_nl    = texio.write_nl
287 local flush_list         = node.flush_list

288 local luatexbase_warning
289 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.

```

290 local modules = modules or { }

```

(End definition for modules.)

`provides_module` Local function to write to the log.

```
291 local function luatexbase_log(text)
292   texio_write_nl("log", text)
293 end
```

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

```
294 local function provides_module(info)
295   if not (info and info.name) then
296     luatexbase_error("Missing module name for provides_module")
297   end
298   local function spaced(text)
299     return text and (" " .. text) or ""
300   end
301   luatexbase_log(
302     "Lua module: " .. info.name
303     .. spaced(info.date)
304     .. spaced(info.version)
305     .. spaced(info.description)
306   )
307   modules[info.name] = info
308 end
309 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 `LATEX` 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.

```
310 local function msg_format(mod, msg_type, text)
311   local leader = ""
312   local cont
313   local first_head
314   if mod == "LaTeX" then
315     cont = string.gsub(leader, ".", " ")
316     first_head = leader .. "LaTeX: "
317   else
318     first_head = leader .. "Module " .. msg_type
319     cont = "(" .. mod .. ")"
320     .. string.gsub(first_head, ".", " ")
321     first_head = leader .. "Module " .. mod .. " " .. msg_type .. ":" ..
322   end
323   if msg_type == "Error" then
324     first_head = "\n" .. first_head
325   end
326   if string.sub(text,-1) ~= "\n" then
```

```

327     text = text .. " "
328   end
329   return first_head .. " "
330   .. string.gsub(
331     text
332   .. "on input line "
333     .. tex.inputlineno, "\n", "\n" .. cont .. " "
334   )
335   .. "\n"
336 end

module_info Write messages.
module_warning
module_error 337 local function module_info(mod, text)
338   texio_write_nl("log", msg_format(mod, "Info", text))
339 end
340 luatexbase.module_info = module_info
341 local function module_warning(mod, text)
342   texio_write_nl("term and log", msg_format(mod, "Warning", text))
343 end
344 luatexbase.module_warning = module_warning
345 local function module_error(mod, text)
346   error(msg_format(mod, "Error", text))
347 end
348 luatexbase.module_error = module_error

```

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

Dedicated versions for the rest of the code here.

```

349 function luatexbase_warning(text)
350   module_warning("luatexbase", text)
351 end
352 function luatexbase_error(text)
353   module_error("luatexbase", text)
354 end

```

5.12 Accessing register numbers from Lua

Collect up the data from the T_EX level into a Lua table: from version 0.80, LuaT_EX makes that easy.

```

355 local luaregisterbasetable = { }
356 local registermap = {
357   attributezero = "assign_attr"      ,
358   charzero     = "char_given"       ,
359   CountZero    = "assign_int"       ,
360   dimenzero    = "assign_dimen"     ,
361   mathcharzero = "math_given"       ,
362   muskipzero   = "assign_mu_skip"  ,
363   skipzero     = "assign_skip"      ,
364   tokszero    = "assign_toks"      ,
365 }
366 local createtoken
367 if tex.luatexversion > 81 then
368   createtoken = token.create
369 elseif tex.luatexversion > 79 then

```

```

370     createtoken = newtoken.create
371 end
372 local hashtokens      = tex.hashtokens()
373 local luatexversion   = tex.luatexversion
374 for i,j in pairs (registermap) do
375     if luatexversion < 80 then
376         luaregisterbasetable[hashtokens[i][1]] =
377             hashtokens[i][2]
378     else
379         luaregisterbasetable[j] = createtoken(i).mode
380     end
381 end

```

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

```

382 local registernumber
383 if luatexversion < 80 then
384     function registernumber(name)
385         local nt = hashtokens[name]
386         if(nt and luaregisterbasetable[nt[1]]) then
387             return nt[2] - luaregisterbasetable[nt[1]]
388         else
389             return false
390         end
391     end
392 else
393     function registernumber(name)
394         local nt = createtoken(name)
395         if(luaregisterbasetable[nt.cmdname]) then
396             return nt.mode - luaregisterbasetable[nt.cmdname]
397         else
398             return false
399         end
400     end
401 end
402 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.

```

403 local attributes=setmetatable(
404 {}, {
405     __index = function(t,key)
406         return registernumber(key) or nil
407     end}
408 )
409 luatexbase.attributes = attributes

```

```

411 local attribute_count_name =
412         attribute_count_name or "e@alloc@attribute@count"
413 local function new_attribute(name)
414     tex_setcount("global", attribute_count_name,
415                 tex_count[attribute_count_name] + 1)
416     if tex_count[attribute_count_name] > 65534 then
417         luatexbase_error("No room for a new \\attribute")
418     end
419     attributes[name]= tex_count[attribute_count_name]
420     luatexbase_log("Lua-only attribute " .. name .. " = " ..
421                     tex_count[attribute_count_name])
422     return tex_count[attribute_count_name]
423 end
424 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.

```

425 local whatsit_count_name = whatsit_count_name or "e@alloc@whatsit@count"
426 local function new_whatsit(name)
427     tex_setcount("global", whatsit_count_name,
428                 tex_count[whatsit_count_name] + 1)
429     if tex_count[whatsit_count_name] > 65534 then
430         luatexbase_error("No room for a new custom whatsit")
431     end
432     luatexbase_log("Custom whatsit " .. (name or "") .. " = " ..
433                     tex_count[whatsit_count_name])
434     return tex_count[whatsit_count_name]
435 end
436 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.

```

437 local bytecode_count_name =
438         bytecode_count_name or "e@alloc@bytecode@count"
439 local function new_bytecode(name)
440     tex_setcount("global", bytecode_count_name,
441                 tex_count[bytecode_count_name] + 1)
442     if tex_count[bytecode_count_name] > 65534 then
443         luatexbase_error("No room for a new bytecode register")
444     end
445     luatexbase_log("Lua bytecode " .. (name or "") .. " = " ..
446                     tex_count[bytecode_count_name])
447     return tex_count[bytecode_count_name]
448 end
449 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.

```
450 local chunkname_count_name =
451         chunkname_count_name or "e@alloc@luachunk@count"
452 local function new_chunkname(name)
453     tex_setcount("global", chunkname_count_name,
454                 tex_count[chunkname_count_name] + 1)
455     local chunkname_count = tex_count[chunkname_count_name]
456     chunkname_count = chunkname_count + 1
457     if chunkname_count > 65534 then
458         luatexbase_error("No room for a new chunkname")
459     end
460     lua.name[chunkname_count]=name
461     luatexbase_log("Lua chunkname " .. (name or "") .. " = " ..
462                     chunkname_count .. "\n")
463     return chunkname_count
464 end
465 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.

```
466 local luafunction_count_name =
467         luafunction_count_name or "e@alloc@luafunction@count"
468 local function new_luafunction(name)
469     tex_setcount("global", luafunction_count_name,
470                 tex_count[luafunction_count_name] + 1)
471     if tex_count[luafunction_count_name] > 65534 then
472         luatexbase_error("No room for a new luafunction register")
473     end
474     luatexbase_log("Lua function " .. (name or "") .. " = " ..
475                     tex_count[luafunction_count_name])
476     return tex_count[luafunction_count_name]
477 end
478 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.

```
479 local callbacklist = callbacklist or {}
```

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

```
480 local list, data, exclusive, simple, reverselist = 1, 2, 3, 4, 5
481 local types    = {
482     list      = list,
483     data      = data,
484     exclusive = exclusive,
485     simple    = simple,
486     reverselist = reverselist,
487 }
```

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.)

```
488 local callbacktypes = callbacktypes or {
```

Section 8.2: file discovery callbacks.

```
489   find_read_file      = exclusive,
490   find_write_file     = exclusive,
491   find_font_file      = data,
492   find_output_file    = data,
493   find_format_file    = data,
494   find_vf_file        = data,
495   find_map_file       = data,
496   find_enc_file       = data,
497   find_pk_file        = data,
498   find_data_file      = data,
499   find_opentype_file  = data,
500   find_truetype_file  = data,
501   find_type1_file     = data,
502   find_image_file     = data,

503   open_read_file      = exclusive,
504   read_font_file      = exclusive,
505   read_vf_file        = exclusive,
506   read_map_file       = exclusive,
507   read_enc_file       = exclusive,
508   read_pk_file        = exclusive,
509   read_data_file      = exclusive,
510   read_truetype_file  = exclusive,
511   read_type1_file     = exclusive,
512   read_opentype_file  = exclusive,
```

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

```
513  find_cidmap_file    = data,  
514  read_cidmap_file   = exclusive,
```

Section 8.3: data processing callbacks.

```
515  process_input_buffer = data,  
516  process_output_buffer = data,  
517  process_jobname      = data,
```

Section 8.4: node list processing callbacks.

```
518  contribute_filter     = simple,  
519  buildpage_filter     = simple,  
520  build_page_insert    = exclusive,  
521  pre_linebreak_filter = list,  
522  linebreak_filter     = exclusive,  
523  append_to_vlist_filter = exclusive,  
524  post_linebreak_filter = reverselist,  
525  hpack_filter         = list,  
526  vpack_filter         = list,  
527  hpack_quality        = list,  
528  vpack_quality        = list,  
529  pre_output_filter    = list,  
530  process_rule          = exclusive,  
531  hyphenate             = simple,  
532  ligaturing            = simple,  
533  kerning               = simple,  
534  insert_local_par     = simple,  
535  pre_mlist_to_hlist_filter = list,  
536  mlist_to_hlist        = exclusive,  
537  post_mlist_to_hlist_filter = reverselist,  
538  new_graf               = exclusive,
```

Section 8.5: information reporting callbacks.

```
539  pre_dump              = simple,  
540  start_run              = simple,  
541  stop_run               = simple,  
542  start_page_number     = simple,  
543  stop_page_number       = simple,  
544  show_error_hook        = simple,  
545  show_warning_message  = simple,  
546  show_error_message    = simple,  
547  show_lua_error_hook   = simple,  
548  start_file             = simple,  
549  stop_file              = simple,  
550  call_edit              = simple,  
551  finish_synctex        = simple,  
552  wrapup_run             = simple,
```

Section 8.6: PDF-related callbacks.

```
553  finish_pdffile         = data,  
554  finish_pdfpage        = data,  
555  page_objnum_provider  = data,  
556  page_order_index      = data,  
557  process_pdf_image_content = data,
```

Section 8.7: font-related callbacks.

```

558     define_font          = exclusive,
559     glyph_info           = exclusive,
560     glyph_not_found      = exclusive,
561     glyph_stream_provider = exclusive,
562     make_extensible      = exclusive,
563     font_descriptor_objnum_provider = exclusive,
564     input_level_string    = exclusive,
565     provide_charproc_data = exclusive,
566 }
567 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.

```

568 local callback_register = callback_register or callback.register
569 function callback.register()
570   luatexbase_error("Attempt to use callback.register() directly\n")
571 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.

```
572 local function data_handler(name)
573   return function(data, ...)
574     for _,i in ipairs(callbacklist[name]) do
575       data = i.func(data,...)
576     end
577     return data
578   end
579 end
```

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

```
580 local function data_handler_default(value)
581   return value
582 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.

```
583 local function exclusive_handler(name)
584   return function(...)
585     return callbacklist[name][1].func(...)
586   end
587 end
```

Handler for **list** callbacks.

```
588 local function list_handler(name)
589   return function(head, ...)
590     local ret
591     local alltrue = true
592     for _,i in ipairs(callbacklist[name]) do
593       ret = i.func(head, ...)
594       if ret == false then
595         luatexbase_warning(
596           "Function '" .. i.description .. "' returned false\n"
597           .. "in callback '" .. name .. "'")
598       )
599       return false
600     end
601     if ret ~= true then
602       alltrue = false
603       head = ret
604     end
605   end
606   return alltrue and true or head
607 end
608 end
```

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

```
609 local function list_handler_default()
610   return true
611 end
```

Handler for `reverselist` callbacks.

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

Handler for `simple` callbacks.

```
635 local function simple_handler(name)
636   return function(...)
637     for _,i in ipairs(callbacklist[name]) do
638       i.func(...)
639     end
640   end
641 end
```

Default for user-defined `simple` callbacks without explicit default.

```
642 local function simple_handler_default()
643 end
```

Keep a handlers table for indexed access and a table with the corresponding default functions.

```
644 local handlers  = {
645   [data]      = data_handler,
646   [exclusive] = exclusive_handler,
647   [list]      = list_handler,
648   [reverselist] = reverselist_handler,
649   [simple]    = simple_handler,
650 }
651 local defaults = {
652   [data]      = data_handler_default,
653   [exclusive] = nil,
654   [list]      = list_handler_default,
655   [reverselist] = list_handler_default,
656   [simple]    = simple_handler_default,
657 }
```

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.

```
658 local user_callbacks_defaults = {
659   pre_mlist_to_hlist_filter = list_handler_default,
660   mlist_to_hlist = node.mlist_to_hlist,
661   post_mlist_to_hlist_filter = list_handler_default,
662 }
```

`create_callback` The allocator itself.

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

(End definition for `create_callback`.)

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

```
688 local function call_callback(name,...)
689   if not name or name == "" then
690     luatexbase_error("Unable to create callback:\n" ..
691                      "valid callback name required")
692   end
693   if user_callbacks_defaults[name] == nil then
694     luatexbase_error("Unable to call callback '" .. name ..
695                      "'\nunknown or empty")
696   end
697   local l = callbacklist[name]
698   local f
699   if not l then
700     f = user_callbacks_defaults[name]
```

```

701     else
702         f = handlers[callbacktypes[name]](name)
703     end
704     return f(...)
705 end
706 luatexbase.call_callback=call_callback
(End definition for call_callback.)

```

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

```

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

```

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

```

722     local l = callbacklist[name]
723     if l == nil then
724         l = { }
725         callbacklist[name] = l

```

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

```

726     if user_callbacks_defaults[name] == nil then
727         callback_register(name, handlers[callbacktypes[name]](name))
728     end
729 end

```

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

```

730     local f = {
731         func      = func,
732         description = description,
733     }
734     local priority = #l + 1
735     if callbacktypes[name] == exclusive then
736         if #l == 1 then
737             luatexbase_error(
738                 "Cannot add second callback to exclusive function\n" ..
739                 name .. "'")
740         end
741     end
742     table.insert(l, priority, f)

```

Keep user informed.

```
743     luatexbase_log(
744         "Inserting '' .. description .. '' at position "
745         .. priority .. " in '' .. name .. ''."
746     )
747 end
748 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.

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

```
768     local index = false
769     for i,j in ipairs(l) do
770         if j.description == description then
771             index = i
772             break
773         end
774     end
775     if not index then
776         luatexbase_error(
777             "No callback '' .. description .. '' registered for '' ..
778             name .. ''\n")
779     end
780     local cb = l[index]
781     table.remove(l, index)
782     luatexbase_log(
783         "Removing '' .. description .. '' from '' .. name .. ''."
784     )
785     if #l == 0 then
786         callbacklist[name] = nil
787         if user_callbacks_defaults[name] == nil then
788             callback_register(name, nil)
```

```

789     end
790   end
791   return cb.func,cb.description
792 end
793 luatexbase.remove_from_callback = remove_from_callback

(End definition for remove_from_callback.)

```

in_callback Look for a function description in a callback.

```

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

```

810 local function disable_callback(name)
811   if(callbacklist[name] == nil) then
812     callback_register(name, false)
813   else
814     luatexbase_error("Callback list for " .. name .. " not empty")
815   end
816 end
817 luatexbase.disable_callback = disable_callback

```

(*End definition for disable_callback.*)

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

```

818 local function callback_descriptions (name)
819   local d = {}
820   if not name
821     or name == ""
822     or not callbacklist[name]
823     or not callbacktypes[name]
824     then
825       return d
826     else
827       for k, i in pairs(callbacklist[name]) do
828         d[k]= i.description
829       end
830     end
831   return d

```

```

832 end
833 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!

```

834 local function uninstall()
835   module_info(
836     "luatexbase",
837     "Uninstalling kernel luatexbase code"
838   )
839   callback.register = callback_register
840   luatexbase = nil
841 end
842 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.

```

843 callback_register("mlist_to_hlist", function(head, display_type, need_penalties)
844   local current = call_callback("pre_mlist_to_hlist_filter", head, display_type, need_penalties)
845   if current == false then
846     flush_list(head)
847     return nil
848   elseif current == true then
849     current = head
850   end
851   current = call_callback("mlist_to_hlist", current, display_type, need_penalties)
852   local post = call_callback("post_mlist_to_hlist_filter", current, display_type, need_penalties)
853   if post == true then
854     return current
855   elseif 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 `\catcode`@=\etacatcode\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}{\fi}%
118 〈latexrelease〉      \ifdefinable{filesize}{\fi}%
119 〈latexrelease〉      \ifdefinable{luatexversion}{\ifnum\luatexversion>94\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

```

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.

```

132 〈latexrelease〉\IncludeInRelease{2020/10/01}{\Expl@cs@to@str@@N}%
133 〈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.

```

134 \ExplSyntaxOn
135 \cs_gset_eq:NN \Expl@cs@to@str@@N \cs_to_str:N
136 \cs_gset_eq:NN \Expl@str@if@eq@@nnTF \str_if_eq:nnTF
137 \cs_gset_eq:NN \Expl@cs@prefix@spec@@N \cs_prefix_spec:N
138 \cs_gset_eq:NN \Expl@cs@argument@spec@@N \cs_argument_spec:N
139 \cs_gset_eq:NN \Expl@cs@replacement@spec@@N \cs_replacement_spec:N

```

```

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

```

Here we can't assume that `expl3` is available. It will be if we roll back but if this code is executed rolling forward it needs to be pure 2e.

```

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

```

File f

ltdefns.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 `\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 \@depth \@width \@minus \@plus The following definitions save token space. E.g., using \@height instead of height saves 5 tokens at the cost in time of one macro expansion.
²⁶ \def\@height{height} \def\@depth{depth} \def\@width{width}
²⁷ \def\@minus{minus}
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 {\langle NAME\rangle}
Expands to \def{\langle NAME\rangle}, except name can contain any characters.

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

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

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

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

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

\@ifdefinable {\NAME}{\langle YES\rangle} Executes \langle YES\rangle 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 *{\langle FOO\rangle}{\langle i\rangle}{\langle TEXT\rangle}
User command to define \FOO to be a macro with i arguments (i = 0 if missing) having the definition \langle TEXT\rangle. Produces an error if \FOO already defined.

\renewcommand *{\langle FOO\rangle}{\langle i\rangle}{\langle TEXT\rangle}

```

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

\renewenvironment{FOO}{{i}{DEF1}{DEF2}}
  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 {\i{message}}
  Produces a warning message on the terminal.
  \typein {\i{message}}
  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{FOO}{{i}{DEF1}{DEF2}}
 50 \def\@namedef#1{\expandafter\def\csname #1\endcsname}

  (End definition for \@namedef.)

\@nameuse{FOO}{{i}{DEF1}{DEF2}}
 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.

\@@ifdefinable 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\iffloor creates \foottrue,
\foofalse to go with \iffloor.
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
  \cprotect@
\fi
\protect\_<space>
```

Then if `\protect` is `\@typeset@protect` (normally `\relax`) then we just perform `_<space>`, and otherwise `\cprotect@` 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     \c@latec@info{Redefining \string#1}%
225   \fi\fi
226   \edef\reserved@a{\string#1}%
227   \def\reserved@b{#1}%
228   \edef\reserved@b{\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`.)

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

(End definition for `\c@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 \set@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 \protected@xdef The commands \protected@edef and \protected@xdef perform ‘safe’ \edefs and \xdefs, saving and restoring \protect appropriately. For cases where restoring \protect doesn’t matter, there’s an ‘unsafe’ \unrestored@protected@xdef, useful if you know 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
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> {Undefine \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@newcommandaux#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   \@ifundefined@#1
639   \fi
640   \@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 \l__cmd_args_ii_tl

Hold the modified arguments when dealing with default values or processors.

```
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
\l__cmd_defaults_tl

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_tmpta_tl:w \l__cmd_tmpb_tl

Scratch space.

```
45 \prop_new:N \l__cmd_tmp_prop  
46 \tl_new:N \l__cmd_tmpta_tl  
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:` 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.

`__cmd_defaults_def:`

`__cmd_defaults_def:nn`

`__cmd_defaults_def:nnn`

`__cmd_defaults_aux:`

`__cmd_defaults_error:w`

```

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 type. If any argument is unknown stop the definition.

```

\__cmd_normalize_arg_spec_loop:n
  \cs_new_protected:Npn \__cmd_normalize_arg_spec:n #1
  {
    \int_zero:N \l__cmd_current_arg_int
    \tl_clear:N \l__cmd_last_delimiters_tl
    \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:nxxx { cmd } { processor-in-expandable }
469       { \iow_char:N \\ \l__cmd_function_tl } { \tl_to_str:n {#1} }
470       \__cmd_bad_def:wn
471     }
472     \tl_put_right:Nx \l__cmd_arg_spec_tl { > { \tl_trim_spaces:n {#1} } }
473     \int_decr:N \l__cmd_current_arg_int
474     \bool_set_false:N \l__cmd_grab_expandably_bool
475     \__cmd_normalize_arg_spec_loop:n {#2}
476   }
477 \cs_new_protected:cpx { __cmd_normalize_type_+:w } #1
478   {

```

```

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_no_value_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_ε 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:NnNNNNnnnn`

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:NnNNNNnnnn` 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     \iow_term: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     \iow_term: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 `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 {
 1447 __cmd_grab_E:nnNN {#1} {#2}
 1448 \cs_set_protected:Npn
 1449 __cmd_peek_nonspace_remove:NTF
 1450 }
 1451 \cs_new_protected:Npn __cmd_grab_E_obey_spaces:w #1#2 __cmd_run_code:
 1452 {
 1453 __cmd_grab_E:nnNN {#1} {#2}
 1454 \cs_set_protected_nopar:Npn
 1455 __cmd_peek_meaning_remove:NTF
 1456 }
 1457 \cs_new_protected:Npn __cmd_grab_E_long_obey_spaces:w #1#2 __cmd_run_code:
 1458 {
 1459 __cmd_grab_E:nnNN {#1} {#2}
 1460 \cs_set_protected:Npn
 1461 __cmd_peek_meaning_remove:NTF
 1462 }

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_2:w __cmd_grab_m_3:w __cmd_grab_m_4:w __cmd_grab_m_5:w __cmd_grab_m_6:w __cmd_grab_m_7:w __cmd_grab_m_8:w __cmd_grab_m_9:w __cmd_grab_m_aux:Nnnnnnnnn 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_1:w 1510 \cs_new_protected_nopar:Npn \__cmd_grab_m_aux:Nnnnnnnnn #1#2#3#4#5#6#7#8#9
\__cmd_grab_m_2:w 1511 {
\__cmd_grab_m_3:w 1512     \tl_put_right:No \l__cmd_args_tl
\__cmd_grab_m_4:w 1513     { #1 {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9} }
\__cmd_grab_m_5:w 1514     \l__cmd_signature_tl \__cmd_run_code:
\__cmd_grab_m_6:w 1515 }
\__cmd_grab_m_7:w 1516 \cs_new_protected:cpn { __cmd_grab_m_1:w } #1 \__cmd_run_code:
\__cmd_grab_m_8:w 1517 {
\__cmd_grab_m_9:w 1518     \tl_set:Nn \l__cmd_signature_tl {#1}
\__cmd_grab_m_aux:Nnnnnnnnn 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

```

1600 \tl_new:N \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:NNNo
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

```

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:` `__cmd_grab_v_aux_abort:n`

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

The first step is to grab the first token or group. The generic grabbers $\backslash\langle function\rangle_{\sqcup}$ and $\backslash\langle function\rangle_{\sqcup}$ 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 *<grabber>* will find an opening delimiter in *<piece 2>*, take the $\backslash q_cmd$ as a second delimiter, and find more material delimited by the closing delimiter in the *<input stream>*. We then move the part before the opening delimiter from *<piece 2>* to *<piece 1>*, and the material taken from the *<input stream>* to the *<piece 2>*. Thus, the argument moves gradually from the *<input stream>* to the *<piece 2>*, then to the *<piece 1>* 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 *<piece 1>* is always equal to the number of closing delimiters in *<piece 2>*. We stop grabbing arguments once the *<piece 2>* contains no opening delimiter any more, hence the balance is reached, and the final argument is *<piece 1>* *<piece 2>*. The indirection via $\backslash__cmd_tmp:w$ 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_nil {piece 1} {more for piece 1}, and we want to concatenate all that, removing \q_nil, and keeping the opening delimiter #2. Simply use \use_ii:nn. Also, #8 is {remainder of piece 2} \ERROR, and #9 is \ERROR {more for piece 2}. We concatenate those, replacing the two \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_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 {pieces}, we are ready to loop. However, we must first see if {piece 2} (here #2) contains any opening delimiter #4. Again, we expand #3, this time removing its whole output with \use_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_none:nnn } #2 \q__cmd #5 #4 \q__cmd #5 }
1787   {
1788     \tl_if_blank:oTF { \use_none:nn #1#2 }
1789     { \__cmd_put_arg_expandable:ow { \use_none:nn #1#2 } }
1790     {
1791       \str_if_eq:eeTF
1792       { \exp_not:o { \use_none:nn #1#2 } }
1793       { { \exp_not:o { \use_iii:nnnn #1#2 \q_nil } } }
1794       { \__cmd_put_arg_expandable:ow { \use_iii:nnnn #1#2 } }
1795       { \__cmd_put_arg_expandable:ow { \use_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_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_i: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.
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_tln
\__cmd_split_list_multi:nN
\__cmd_split_list_single:Nn

```

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.

```

1951 \seq_new:N \l__cmd_split_list_seq
1952 \tl_new:N \l__cmd_split_list_tl
1953 \cs_new_protected:Npn \__cmd_split_list:nn #1#2
1954 {
1955   \tl_if_single:nTF {#1}
1956   {
1957     \token_if_cs:NTF #1
1958     { \__cmd_split_list_multi:nn {#1} {#2} }
1959     { \__cmd_split_list_single:Nn #1 {#2} }
1960   }
1961   { \__cmd_split_list_multi:nn {#1} {#2} }
1962 }
1963 \cs_new_protected:Npn \__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 \__cmd_split_list_multi:nn { nV }
1971 \group_begin:
1972 \char_set_catcode_active:N \^^@ 
1973 \cs_new_protected:Npn \__cmd_split_list_single:Nn #1#2
1974 {
1975   \tl_set:Nn \l__cmd_split_list_tl {#2}
1976   \group_begin:
1977   \char_set_lccode:nn { '\^^@ } { '#1 }
1978   \tex_lowercase:D
1979   {
1980     \group_end:
1981     \tl_replace_all:Nnn \l__cmd_split_list_tl { ^@ }
1982   } {#1}
1983   \__cmd_split_list_multi:nV {#1} \l__cmd_split_list_tl
1984 }
1985 \group_end:

```

(End definition for `__cmd_split_list:nn`, `__cmd_split_list_multi:nn`, and `__cmd_split_list_single:Nn`.)

```

\__cmd_split_argument:nnn
  \cmd_split_argument_aux:nnnn
\__cmd_split_argument_aux:n
\__cmd_split_argument_aux:wn

```

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 \__cmd_split_argument:nnn #1#2#3
1987 {
1988   \__cmd_split_list:nn {#2} {#3}
1989   \exp_args:Nf \__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  {  
        \exp_after:wN \exp_after:wN  
        \exp_after:wN \__cmd_t1_mapthread_loop:w  
        \exp_after:wN \exp_after:wN  
        \exp_after:wN #3  
        \exp_after:wN #1  
        \exp_after:wN \q_recursion_tail  
        \exp_after:wN \q_mark  
        #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 _cmd_peek_nonspace_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   \_cmd_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   \_cmd_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 }
2224 \cs_new_protected:Npn \_cmd_peek_cs_check_equal:NNN #1#2#3
2225 {
```

```

2226     \str_if_eq:nTF {#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 xpars...
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 } { xpars-arg-type }
2404 { Invalid-argument-type-'#2'~in~#1~(requires~xpars). }
2405 {
2406   The~letter~'#2'~specifies~a~known~but~deprecated~argument~type.~
2407   If~you~really~need~it~you~have~to~load~the~xpars~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.
\BooleanTrue 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 2505 \cs_new_protected:Npn \NewDocumentCommand #1#2#3
\RenewDocumentCommand
\ProvideDocumentCommand
\DeclareDocumentCommand 2506 {
  \__cmd_check_definable:nNT {#1} \NewDocumentCommand
  {
    \cs_if_exist:NTF {#1}
    {
      \msg_error:nnxx { cmd } { already-defined }
      { \use:nnn \token_to_str:N {#1} { } }
      { \token_to_str:N \NewDocumentCommand }
    }
    { \__cmd_declare_cmd:Nnn {#1} {#2} {#3} }
  }
}
\cs_new_protected:Npn \RenewDocumentCommand #1#2#3
{
  
```

```

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.)

\ProcessedArgument Processed arguments are returned using this name, which is reserved here although the definition will change.

```
2625 \tl_new:N \ProcessedArgument
```

(End definition for \ProcessedArgument.)

\ReverseBoolean Simple copies.

```

\SplitArgument 2626 \cs_new_eq:NN \ReverseBoolean \__cmd_bool_reverse:N
  \SplitList 2627 \cs_new_eq:NN \SplitArgument \__cmd_split_argument:nnn
\TrimSpaces 2628 \cs_new_eq:NN \SplitList \__cmd_split_list:nn
2629 \cs_new_eq:NN \TrimSpaces \__cmd_trim_spaces:n

```

(End definition for \ReverseBoolean and others.)

\ProcessList To support \SplitList.

```
2630 \cs_new_eq:NN \ProcessList \tl_map_function:nN
```

(End definition for \ProcessList.)

```

\GetDocumentCommandArgSpec More simple mappings, with a check that the argument is a single control sequence or
    \GetDocumentEnvironmentArgSpec active character.

\ShowDocumentCommandArgSpec 2631 \cs_new_protected:Npn \GetDocumentCommandArgSpec #1
    \ShowDocumentEnvironmentArgSpec 2632 {
        \cmd_check_definable:nNT {#1} \GetDocumentCommandArgSpec
        { \cmd_get_arg_spec:N #1 }
    }
2636 \cs_new_eq:NN \GetDocumentEnvironmentArgSpec \cmd_get_arg_spec:n
2637 \cs_new_protected:Npn \ShowDocumentCommandArgSpec #1
    \ShowDocumentEnvironmentArgSpec 2638 {
        \cmd_check_definable:nNT {#1} \ShowDocumentCommandArgSpec
        { \cmd_show_arg_spec:N #1 }
    }
2642 \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:
2643 ⟨latexrelease⟩\cs_gset_eq:NN \kernel_chk_if_free_cs:N \cmd_chk_if_free_cs:N
2644 ⟨latexrelease⟩\cs_undefine:N \cmd_chk_if_free_cs:N
2645 ⟨latexrelease⟩
2646 ⟨latexrelease⟩\IncludeInRelease{0000/00/00}{ltcmand}%
2647 ⟨latexrelease⟩
2648 ⟨latexrelease⟩
2649 ⟨latexrelease⟩\EndModuleRelease
2650 \ExplSyntaxOff
2651 ⟨latexrelease⟩\@ifundefined{ExplSyntaxOff}{}{\@latexrelease@postexpl}
2652 ⟨latexrelease⟩\catcode`^=\@latexrelease@catcode@null\relax
2653 ⟨/2ekernel | latexrelease⟩

```

We need to stop DocStrip treating @@ in a special way at this point.

```

2654 ⟨@@=⟩

```

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 {⟨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 {<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 {<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 {<default label>}
  <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 {<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 {<hook>}{{<label1>}}{<relation>}{{<label2>}}
```

Defines a relation between `<label1>` and `<label2>` for a given `<hook>`. If `<hook>` is `??` this defines a default relation for all hooks that use the two labels, i.e., that have chunks of code labeled with `<label1>` and `<label2>`. Rules specific to a given hook take precedence over default rules that use `??` as the `<hook>`.

Currently, the supported relations are the following:

`before` or `<` Code for `<label1>` comes before code for `<label2>`.

`after` or `>` Code for `<label1>` comes after code for `<label2>`.

`incompatible-warning` Only code for either `<label1>` or `<label2>` 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 `<label1>` overwrites code for `<label2>`. More precisely, code for `<label2>` 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 `<label1>` and `<label2>` 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 `<hook>` and `<label>` 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 `\label1` and `\label2` are unrelated for the given `\hook`.

`\DeclareDefaultHookRule` `\DeclareDefaultHookRule{\label1}{\relation}{\label2}`

This sets up a relation between `\label1` and `\label2` 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 `\label` 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 `\hook` 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 `\truecode` or `\falsecode` depending on the result.

The `\hook` 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`

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

-> $\langle next\text{-}code \rangle$

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{\label}{[example-hook]}`.

Lines 11 and 12 show the rules declared that affect this hook in the format

$\langle label-1 \rangle | \langle label-2 \rangle$ with $\langle default? \rangle$ relation $\langle relation \rangle$

which means that the $\langle relation \rangle$ applies to $\langle label-1 \rangle$ and $\langle label-2 \rangle$, in that order, as detailed in `\DeclareHookRule`. If the relation is `default` it means that this rule applies to $\langle label-1 \rangle$ and $\langle label-2 \rangle$ 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`
`\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 _{ε} 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.

`\hook_new:n`
`\hook_new_reversed:n`
`\hook_new_pair:nn`

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`

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.

<code>\hook_activate_generic:n</code>	<code>\hook_activate_generic:n {<hook>}</code>
	This is like <code>\hook_new:n</code> but it does nothing if the hook was previously declared with <code>\hook_new:n</code> . 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 <code>\hook_new:n</code> should be used instead of this.
<code>\hook_use:n</code>	<code>\hook_use:n {<hook>}</code>
	Executes the <code>{<hook>}</code> code followed (if set up) by the code for next invocation only, then empties that next invocation code.
	The <code><hook></code> cannot be specified using the dot-syntax. A leading <code>.</code> is treated literally.
<code>\hook_use_once:n</code>	<code>\hook_use_once:n {<hook>}</code>
	Changes the <code>{<hook>}</code> status so that from now on any addition to the hook code is executed immediately. Then execute any <code>{<hook>}</code> code already set up.
	The <code><hook></code> cannot be specified using the dot-syntax. A leading <code>.</code> is treated literally.
<code>\hook_gput_code:nnn</code>	<code>\hook_gput_code:nnn {<hook>} {<label>} {<code>}</code>
	Adds a chunk of <code><code></code> to the <code><hook></code> labeled <code><label></code> . If the label already exists the <code><code></code> is appended to the already existing code.
	If code is added to an external <code><hook></code> (of the kernel or another package) then the convention is to use the package name as the <code><label></code> not some internal module name or some other arbitrary string.
	The <code><hook></code> and <code><label></code> can be specified using the dot-syntax to denote the current package name. See section 2.1.5.
<code>\hook_gput_next_code:nn</code>	<code>\hook_gput_next_code:nn {<hook>} {<code>}</code>
	Adds a chunk of <code><code></code> for use only in the next invocation of the <code><hook></code> . Once used it is gone.
	This is simpler than <code>\hook_gput_code:nnn</code> , 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><code></code> .
	The <code><hook></code> can be specified using the dot-syntax to denote the current package name. See section 2.1.5.
<code>\hook_gclear_next_code:n</code>	<code>\hook_gclear_next_code:n {<hook>}</code>
	Undo any earlier <code>\hook_gput_next_code:nn</code> .

\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_if_empty:nTF {<hook>} {<true code>} {<false code>}
```

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_log:n

```
\hook_show: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_off:

```
\hook_debug_on:
```

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 _{ε} interfaces

L^AT_EX 2 _{ε} 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.

There is also `\AtBeginDvi` which is discussed in conjunction with the shipout hooks.

The few hooks that existed previously in L^AT_EX 2 _{ε} 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.

⁸As with everything in TeX there is not 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!”

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.

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 “`<type>/<name>/<position>`”, where `<type>` is from the predefined list shown below, and `<name>` is the variable part whose meaning will depend on the `<type>`. The last component, `<position>`, 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 – `<name>` is the name of the environment, and available values for `<position>` are `before`, `begin`, `end`, and `after`;
- cmd** Hooks added to and executed before and after commands – `<name>` is the name of the command, and available values for `<position>` are `before` and `after`;
- file** Hooks executed before and after reading a file – `<name>` is the name of the file (with extension), and available values for `<position>` are `before` and `after`;
- package** Hooks executed before and after loading packages – `<name>` is the name of the package, and available values for `<position>` are `before` and `after`;
- class** Hooks executed before and after loading classes – `<name>` is the name of the class, and available values for `<position>` are `before` and `after`;
- include** Hooks executed before and after `\included` files – `<name>` is the name of the included file (without the `.tex` extension), and available values for `<position>` 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 `<env>` has now four associated hooks coming with it:

- env/<env>/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/⟨env⟩/begin This hook is executed as part of `\begin` 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/⟨env⟩/end This hook is executed as part of `\end` 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 `\langle env \rangle` 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.

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.

3.1.2 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`.

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_E 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 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 `1tshipout-doc.pdf` or with code in `1tshipout-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 `1tpara-doc.pdf` or with code in `1tpara-code.pdf`.

3.6 Hooks provided in NFSS commands

In languages that need to support 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 188.)

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

`\l__hook_tmpa_bool` Scratch boolean used throughout the package.

```

24 \bool_new:N \l__hook_tmpa_bool

```

(End definition for `\l__hook_tmpa_bool`.)

`\l__hook_return_tl` Scratch variables used throughout the package.

```

25 \tl_new:N \l__hook_return_tl

```

```

26 \tl_new:N \l__hook_tmpa_tl

```

```

27 \tl_new:N \l__hook_tmpb_tl

```

(End definition for `\l__hook_return_tl`, `\l__hook_tmpa_tl`, and `\l__hook_tmpb_tl`.)

`\g__hook_all_seq` In a few places we need a list of all hook names ever defined so we keep track if them in this sequence.

```

28 \seq_new:N \g__hook_all_seq

```

(End definition for `\g__hook_all_seq`.)

`\l__hook_cur_hook_tl` Stores the name of the hook currently being sorted.

```

29 \tl_new:N \l__hook_cur_hook_tl

```

(End definition for `\l__hook_cur_hook_tl`.)

`\l__hook_work_prop` A property list holding a copy of the `\g__hook_<hook>_code_prop` of the hook being sorted to work on, so that changes don't act destructively on the hook data structure.

```

30 \prop_new:N \l__hook_work_prop

```

(End definition for `\l__hook_work_prop`.)

`\g__hook_used_prop` All hooks that receive code (for use in debugging display).

```

31 \prop_new:N \g__hook_used_prop

```

(End definition for `\g__hook_used_prop`.)

`\g__hook_hook_curr_name_tl` Default label used for hook commands, and a stack to keep track of packages within packages.

```

32 \tl_new:N \g__hook_hook_curr_name_tl

```

```

33 \seq_new:N \g__hook_name_stack_seq

```

(End definition for `\g__hook_hook_curr_name_tl` and `\g__hook_name_stack_seq.`)

`__hook_tmp:w` Temporary macro for generic usage.
 34 `\cs_new_eq:NN __hook_tmp:w ?`

(End definition for `__hook_tmp:w.`)

`\tl_gremove_once:Nx` Some variants of `expl3` functions.
`\tl_show:x` *FMi: should probably be moved to `expl3`*
`\tl_log:x`

35 `\cs_generate_variant:Nn \tl_gremove_once:Nn { Nx }`
 36 `\cs_generate_variant:Nn \tl_show:n { x }`
 37 `\cs_generate_variant:Nn \tl_log:n { x }`

(End definition for `\tl_gremove_once:Nx`, `\tl_show:x`, and `\tl_log:x.`)

`\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` 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 Same as above.
\_\_hook_tl_gput_right:Nn
\_\_hook_tl_gput_right:cn
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 Same as above.
\_\_hook_tl_gput_left:No
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
\_\_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

Hooks have a name (called *<hook>* in the description below) and for each hook we have to provide a number of data structures. These are

\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>_t1 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 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 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 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 186.)

`__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., `\@beginocumenthook`. 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
`_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 [186](#).)

\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 [186](#).)

`_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 <|latexrelease>\EndIncludeInRelease
139 <|latexrelease>\IncludeInRelease{2020/10/01}
140 <|latexrelease>          {\hook_disable_generic:n}{Disable~hooks}
141 <|latexrelease>
142 <|latexrelease>\cs_new_protected:Npn \hook_disable_generic:n #1 {}
143 <|latexrelease>
144 <|latexrelease>\EndIncludeInRelease
```

(End definition for `\hook_disable_generic:n`, `__hook_disable:n`, and `__hook_if_disabled:nTF`. This function is documented on page [186](#).)

`\hook_activate_generic:n`
`__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 <|latexrelease>\IncludeInRelease{2021/06/01}%
146 <|latexrelease>          {\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 187.)

```

163 \end{macro}
164 \end{macro}
165 \end{macro}
166 \end{macro}
167 \end{macro}
168 \end{macro}
169 \end{macro}

```

4.5 Parsing a label

__hook_parse_label_default:n

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     { \__hook_currname_or_default: }
174     { \tl_trim_spaces_apply:nN {#1} \__hook_parse_dot_label:n }
175 }

```

(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.

```

176 \cs_new:Npn \__hook_parse_dot_label:n #1
177 {
178     \tl_if_empty:nTF {#1}
179     {
180         \msg_expandable_error:nn { hooks } { empty-label }
181         \__hook_currname_or_default:
182     }

```

```

183     {
184         \str_if_eq:nNF {#1} { . }
185         { \__hook_curname_or_default: }
186         { \__hook_parse_dot_label:w #1 ./ \s__hook_mark }
187     }
188 }
189 \cs_new:Npn \__hook_parse_dot_label:w #1 ./ #2 \s__hook_mark
190 {
191     \tl_if_empty:nTF {#1}
192     { \__hook_parse_dot_label_aux:w #2 \s__hook_mark }
193     {
194         \tl_if_empty:nTF {#2}
195         { \__hook_make_name:n {#1} }
196         { \__hook_parse_dot_label_cleanup:w #1 ./ #2 \s__hook_mark }
197     }
198 }
199 \cs_new:Npn \__hook_parse_dot_label_cleanup:w #1 ./ \s__hook_mark {#1}
200 \cs_new:Npn \__hook_parse_dot_label_aux:w #1 ./ \s__hook_mark
201 { \__hook_curname_or_default: / \__hook_make_name:n {#1} }

```

(End definition for `__hook_parse_dot_label:n` and others.)

`__hook_curname_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_curname_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             { Empty~default~label. }
210             \__hook_make_name:n { label-missing }
211         }
212         { \@currname }
213     }
214     { \g__hook_hook_curr_name_tl }
215 }

```

(End definition for `__hook_curname_or_default:..`)

`__hook_make_name:n`
`__hook_make_name:w`

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`.)

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:Nn
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:Nnn #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:Nnnn #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.)

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 `latexrelease` we push something on the stack to support roll forward. But in some rare cases, `latexrelease` may be loaded inside another package (notably `platexrelease`), so we'll first push the **top-level** entry:

```

256 <|latexrelease>\seq_if_empty:NT \g__hook_name_stack_seq
257 <|latexrelease> { \seq_gput_right:Nn \g__hook_name_stack_seq { top-level } }
then we dissect the \currnamestack, adding \currname to the stack:
```

```

258 <|latexrelease>\cs_set_protected:Npn \__hook_tmp:w #1 #2 #3
259 <|latexrelease> {
260 <|latexrelease> \quark_if_recursion_tail_stop:n {#1}
261 <|latexrelease> \seq_gput_right:Nn \g__hook_name_stack_seq {#1}
262 <|latexrelease> \__hook_tmp:w
263 <|latexrelease> }
264 <|latexrelease>\exp_after:wN \__hook_tmp:w \currnamestack
265 <|latexrelease> \q_recursion_tail \q_recursion_tail
266 <|latexrelease> \q_recursion_tail \q_recursion_stop
```

and finally set the default label to be the \currname:

```

267 <|latexrelease>\tl_gset:Nx \g__hook_hook_curr_name_tl { \currname }
268 <|latexrelease>\seq_gpop_right:NN \g__hook_name_stack_seq \l__hook_tma_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_t1
290 { \tl_gset_eq:NN \g__hook_hook_curr_name_tl \l__hook_return_t1 }
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   }
305
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_gput_code:do:nnn

```

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:nnn { hooks } { hook-disabled } {#1} }
338             { \_\_hook_try_declarngeneric_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_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 187.)

`_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 <latexrelease>\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 <latexrelease>\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_declarng_generic_hook:nnn` and
`_hook_try_declarng_generic_next_hook:nn`)

`_hook_try_declarng_generic_hook:nNnn`
`_hook_try_declarng_generic_hook_split:nNnn`

`_hook_try_declarng_generic_hook:nNnn` 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_declarng_generic_hook:nNnn #1
401 <latexrelease> {
402 <latexrelease>   \_hook_if_file_hook:wTF #1 / \s_hook_mark
403 <latexrelease>   {
404 <latexrelease>     \exp_args:Ne \_hook_try_declarng_generic_hook_split:nNnn
405 <latexrelease>     { \exp_args:Ne \_hook_file_hook_normalize:n {#1} }
406 <latexrelease>   }
407 <latexrelease>   { \_hook_try_declarng_generic_hook_split:nNnn {#1} }
408 <latexrelease> }

409 <latexrelease> \cs_new_protected:Npn \_hook_try_declarng_generic_hook_split:nNnn #1 #2 #3
410 <latexrelease> {
411 <latexrelease>   \_hook_try_declarng_generic_hook:wnTF #1 // \scan_stop: {#1}
412 <latexrelease>   { #2 }
413 <latexrelease>   { #3 } {#1}
414 <latexrelease> }
415 <latexrelease> \EndIncludeInRelease

```

(End definition for `_hook_try_declarng_generic_hook:nNnn` and
`_hook_try_declarng_generic_hook_split:nNnn`)

`_hook_try_declarng_generic_hook:wnTF`

```

416 <latexrelease> \IncludeInRelease{2021/11/15}{\_hook_try_declarng_generic_hook:wn}%
417 <latexrelease>           {Standardise-generic~hook~names}
418 \prg_new_protected_conditional:Npnn \_hook_try_declarng_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 ($\text{\c__hook_generic}_{\langle type \rangle} / . / \langle place \rangle_{_t1}$), 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 $\text{\exp_after:wN} \dots \text{\exp:w}$ trick is there to remove the conditional structure inserted by $\text{_hook_try_declaring_generic_hook:wnTF}$ 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 $\text{\prg_return_false:}$.

```

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 }

```

$\text{__hook_DEPRECATED_GENERIC_WARN:n}$ 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 $\langle name \rangle$ and $\langle place \rangle$ and adding the code to the correct hook.

```

53 \cs_new_protected:Npn \_\_hook_DO_DEPRECATED_GENERIC:Nn #1 #2
54   { \_\_hook_DO_DEPRECATED_GENERIC:Nw #1 #2 \s_\_hook_mark }
55 \cs_new_protected:Npn \_\_hook_DO_DEPRECATED_GENERIC:Nw #
56   #2 / #3 / #4 \s_\_hook_mark
57   { #1 { #2 / #4 / #3 } }
58 \cs_new_protected:Npn \_\_hook_DECLARE_DEPRECATED_GENERIC:NNn #1 #2 #3
59   { \_\_hook_DECLARE_DEPRECATED_GENERIC:NNw #1 #2 #3 \s_\_hook_mark }
60 \cs_new_protected:Npn \_\_hook_DECLARE_DEPRECATED_GENERIC:NNw #1 #2
61   #3 / #4 / #5 \s_\_hook_mark
62   {
63     \_\_hook_TRY_DECLARING_GENERIC_HOOK:wnTF #3 / #5 / #4 / \scan_stop:
64     { #3 / #5 / #4 }
65     #1 #2 { #3 / #5 / #4 }
66   }
67 \EndIncludeInRelease

```

```

468 〈latexrelease〉\IncludeInRelease{2021/06/01}{\_\_hook_try_declarin_generic_hook:wn}
469 〈latexrelease〉                                {Support~cmd-hooks}
470 〈latexrelease〉\prg_new_protected_conditional:Npn \_\_hook_try_declarin_generic_hook:wn
471 〈latexrelease〉    #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
472 〈latexrelease〉    {
473 〈latexrelease〉        \tl_if_empty:nTF {#2}
474 〈latexrelease〉            { \prg_return_false: }
475 〈latexrelease〉            {
476 〈latexrelease〉                \prop_if_in:NnTF \c_\_hook_generics_prop {#1}
477 〈latexrelease〉                    {
478 〈latexrelease〉                        \_\_hook_if_usable:nF {#5}
479 〈latexrelease〉                            {
480 〈latexrelease〉                                \str_if_eq:nnT {#1} { cmd }
481 〈latexrelease〉                                    { \_\_hook_try_put_cmd_hook:n {#5} }
482 〈latexrelease〉                                \_\_hook_make_usable:n {#5}
483 〈latexrelease〉                            }
484 〈latexrelease〉                \prop_if_in:NnTF \c_\_hook_generics_reversed_ii_prop {#2}
485 〈latexrelease〉                    { \tl_gset:cn { g_\_hook_#5_reversed_tl } { - } }
486 〈latexrelease〉                    {
487 〈latexrelease〉                        \prop_if_in:NnT \c_\_hook_generics_reversed_iii_prop {#3}
488 〈latexrelease〉                            { \tl_gset:cn { g_\_hook_#5_reversed_tl } { - } }
489 〈latexrelease〉                        }
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_declarin_generic_hook:wn}%
497 〈latexrelease〉                                {Support~cmd-hooks}
498 〈latexrelease〉\prg_new_protected_conditional:Npn \_\_hook_try_declarin_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_generics_prop {#1}
505 〈latexrelease〉                    {
506 〈latexrelease〉                        \_\_hook_if_declared:nF {#5} { \hook_new:n {#5} }
507 〈latexrelease〉                        \prop_if_in:NnTF \c_\_hook_generics_reversed_ii_prop {#2}
508 〈latexrelease〉                            { \tl_gset:cn { g_\_hook_#5_reversed_tl } { - } }
509 〈latexrelease〉                            {
510 〈latexrelease〉                                \prop_if_in:NnT \c_\_hook_generics_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_declarin_generic_hook:wnTF and others.)

`__hook_if_file_hook:p:w` `__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 <latexrelease>\prg_new_conditional:Npnn \_\_hook\_if\_file\_hook:w
537     #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.)

```

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_genrics_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_#/./before_tl } { + }
575   \tl_const:cn { c__hook_generic_#/./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_#/./before_tl } { }
583   \tl_const:cn { c__hook_deprecated_#/./after_tl } { }
584 }
585 \tl_const:cn { c__hook_deprecated_include./end_tl } { }
586 <|latexrelease>\EndIncludeInRelease

587 <|latexrelease>\IncludeInRelease{2020/10/01}{\c_hook_genrics_prop}%
588 <|latexrelease> {Standardise-generic-hook-names}
589 \prop_const_from_keyval:Nn \c_hook_genrics_prop
590 {cmd=,env=,file=,package=,class=,include=}
591 <|latexrelease>\EndIncludeInRelease

```

(End definition for \c_hook_generic_cmd./before_tl and others.)

```
\c__hook_generics_reversed_ii_prop  
\c__hook_generics_reversed_iii_prop
```

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  
\_\_\_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     { \_\_\_hook_tl_gclear:c { __hook_toplevel~#1 } }  
616     {  
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       { \_\_\_hook_update_hook_code:n {#1} }  
623   }
```

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 188.)

```
\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

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     }
}
```

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 188.)

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 \pdfstrcmp 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>} {<l_A>} {<l_B>} 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:cpn { __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:cpn { __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
\_\_hook_label_ordered:nnTF

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:nnnnn

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:nnn 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:nnnnn #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:nnnnn`.)

`__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:Nn` 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:NV`. 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     {
745         \cs_set_eq:NN \__hook_tl_gput:Nn \__hook_tl_gput_left:Nn
746         \cs_set_eq:NN \__hook_clist_gput:NV \clist_gput_left:NV
747         \cs_set_eq:NN \__hook_tl_gput:Nn \__hook_tl_gput_right:Nn
748         \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` 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_cname:n` and `_hook_seq_cname: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 `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>NNn  
\_hook_initialize_single:ccn
```

`_hook_initialize_single>NNn` 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 `<hook-code-plist>`, `<hook-code-tl>`, `<hook-top-level-code-tl>`, `<hook-next-code-tl>`, `<hook-ordered-labels-clist>` and `<hook-name>` (the latter is only used for debugging—the `<hook-rule-plist>` is accessed using the `<hook-name>`).

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>NNn #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
806     \__hook_clist_gput:NV #2 \l__hook_front_tl
807     \__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         {
811             \int_eval:n
812                 { \cs:w \__hook_tl_cname:n {\#\#1} \cs_end: - 1 }
813         }
814         \int_compare:nNnT
815             { \cs:w \__hook_tl_cname:n {\#\#1} \cs_end: } = 0
816             {
817                 \tl_set:cn { \__hook_tl_cname:n { \l__hook_rear_tl } } {\#\#1}
818                 \tl_set:Nn \l__hook_rear_tl {\#\#1}
819             }
820     }

```

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:Nn`.)

```
\_hook_t1_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_t1_gput:Nn { \ERROR }  
835 \cs_new:Npn \_hook_clist_gput:NV { \ERROR }
```

(End definition for `_hook_t1_gput:Nn` and `_hook_clist_gput:NV`.)

```
\_hook_apply_label_pair:nnn  
\_hook_label_if_exist_apply:nnnF
```

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 `\g_hook_{hook}_rule:#1#2_t1`, and not `\g_hook_{hook}_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 `<label1>`, `<label2>`, `<hook>`, and `<hook-code-plist>`. 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:nnnF` for the `<hook>`, and if no rule is found, also try the `<hook>` 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`
`_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`
`_hook_apply_rule_xW: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 \rightarrow 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.

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.

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}
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} _t1 } ~
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` 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.

```

\hook_log:n
\__hook_log_line:x
\__hook_log_line_indent:x
\__hook_log:nN
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 { 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 } }

```

(End definition for \hook_show:n and others. These functions are documented on page 188.)

This macro takes a *<hook>* and an *<inline function>* and loops through each pair of *<labels>* in the *<hook>*, and if there is a relation between this pair of *<labels>*, the *<inline function>* is executed with #1 = *<relation>*, #2 = *<label₁>* | *<label₂>*, and #3 = *<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 *(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     \ifcsexist: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 187.)

`__hook_gput_next_code:nn`

```

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_declarngeneric_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 #
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 187.)

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 <texrelease>\IncludeInRelease{2021/11/15}{\hook_use:n}
1116 <texrelease>                                {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 <texrelease>\EndIncludeInRelease
1134 <texrelease>\IncludeInRelease{2020/10/01}{\hook_use:n}
1135 <texrelease>                                {Standardise-generic-hook-names}
1136 <texrelease>\cs_new_protected:Npn \hook_use:n #1
1137 <texrelease>  {
1138     <texrelease> \tl_if_exist:cTF { __hook~#1 }
1139     <texrelease>  {
1140         <texrelease> \__hook_preamble_hook:n {#1}
1141         <texrelease> \cs:w __hook~#1 \cs_end:
1142     <texrelease>  }
1143     <texrelease> { \__hook_use:wn #1 / \s__hook_mark {#1} }
1144     <texrelease>  }
1145 <texrelease>\cs_new:Npn \__hook_use_initialized:n #1
1146 <texrelease>  {
1147     <texrelease> \if_cs_exist:w __hook~#1 \cs_end:
1148     <texrelease> \else:
1149     <texrelease>   \__hook_use undefined:w
1150     <texrelease>   \fi:
1151     <texrelease>   \cs:w __hook~#1 \__hook_use_end:
1152     <texrelease>  }
1153 <texrelease>\cs_new:Npn \__hook_use undefined:w #1 #2 __hook~#3 \__hook_use_end:
1154 <texrelease>  {
1155     <texrelease>   #1 % fi
1156     <texrelease>   \__hook_use:wn #3 / \s__hook_mark {#3}
1157     <texrelease>  }
1158 <texrelease>\cs_new_protected:Npn \__hook_preamble_hook:n #1
1159 <texrelease> { \__hook_initialize_hook_code:n {#1} }
1160 <texrelease>\cs_new_eq:NN \__hook_use_end: \cs_end:
1161 <texrelease>\EndIncludeInRelease

```

(End definition for `\hook_use:n` and others. This function is documented on page 187.)

`__hook_use:wn` `__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 ⟨latexrelease⟩\IncludeInRelease{2021/11/15}{\__hook_use:wn}
1163 ⟨latexrelease⟩                                {Standardise-generic-hook-names}
1164 ⟨latexrelease⟩\EndIncludeInRelease
1165 ⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_use:wn}
1166 ⟨latexrelease⟩                                {Standardise-generic-hook-names}
1167 ⟨latexrelease⟩\cs_new:Npn \__hook_use:wn #1 / #2 \s__hook_mark #3
1168 ⟨latexrelease⟩  {
1169 ⟨latexrelease⟩    \str_if_eq:nnTF {#1} { file }
1170 ⟨latexrelease⟩      { \__hook_try_file_hook:n {#3} }
1171 ⟨latexrelease⟩      { } % Hook doesn't exist
1172 ⟨latexrelease⟩  }

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 187.)

`__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 $\langle\text{hook}\rangle$ 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 188.)

`__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.

```
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.

```
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 {
1270     \cs_new:Npn \_hook_if_generic:w #1 / / \s__hook_mark
1271     {
1272         \cs_if_exist:cTF { c__hook_generic_#/./#3_tl }
1273         { \prg_return_true: }
1274         { \prg_return_false: }
1275     }
1276 \prg_new_conditional:Npnn \_hook_if_deprecated_generic:n #1 { T, TF }
1277 {
1278     \cs_new:Npn \_hook_if_deprecated_generic:w #1 / / \s__hook_mark
1279     {
1280         \cs_if_exist:cTF { c__hook_deprecated_#/./#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 {
1290     \cs_new:Npn \_hook_if_generic_reversed:w #1 / / \scan_stop:
1291     \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_tl \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:nnnn { 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:nnnn { 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 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 176.)

```

1399 <|latexrelease>\IncludeInRelease{2021/06/01}%
1400 <|latexrelease>           {\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 177.)

\DisableGenericHook Disabling a generic hook.

```

1402 \NewDocumentCommand \DisableGenericHook { m }{ \hook_disable_generic:n {#1} }

```

(End definition for `\DisableGenericHook`. This function is documented on page 177.)

```

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 178.)

\AddToHookNext

```

1412 \NewDocumentCommand \AddToHookNext { m +m }
1413   { \hook_gput_next_code:nn {#1} {#2} }

```

(End definition for `\AddToHookNext`. This function is documented on page 180.)

\ClearHookNext

```
1414 \NewDocumentCommand \ClearHookNext { m }
1415   { \hook_gclear_next_code:n {#1} }
```

(End definition for \ClearHookNext. This function is documented on page 180.)

\RemoveFromHook

```
1416 \NewDocumentCommand \RemoveFromHook { m o }
1417   { \hook_gremove_code:nn {#1} {#2} }
```

(End definition for \RemoveFromHook. This function is documented on page 179.)

\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

```
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_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 182.)

\UseHook
\UseOneTimeHook

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 177.)

```
\ShowHook
\LogHook 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 185.)

```
\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 186.)

```
\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 183.)

`\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 184.)

`\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:nnnn {#1}{#2}{unrelated}{#3} }
```

(End definition for `\ClearHookRule`. This function is documented on page 184.)

`\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 184.)

`\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 {
\hook_provide_pair:nn 1457   \__hook_DEPRECATED_WARN:nn
\_hook_activate_generic_reversed:n 1458   { hook_disable:n }
\_\_hook_activate_generic_pair:nn 1459   { hook_disable_generic:n }
1460   \hook_disable_generic:n
1461 }
1462 \cs_new_protected:Npn \hook_provide:n
1463 {
\_\_hook_DEPRECATED_WARN:nn 1464   { hook_provide:n }
1465   { hook_activate_generic:n }
1466   \hook_activate_generic:n
1467 }
1468 \cs_new_protected:Npn \hook_provide_reversed:n
1469 {
1470   \__hook_DEPRECATED_WARN:nn
1471   { hook_provide_reversed:n }
1472   { hook_activate_generic:n }
1473   \__hook_activate_generic_reversed:n
1474 }
1475 \cs_new_protected:Npn \hook_provide_pair:nn
1476 {
1477   \__hook_DEPRECATED_WARN:nn
1478   { hook_provide_pair:nn }
1479   { hook_activate_generic:n }
1480   \__hook_activate_generic_pair:nn
1481 }
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 1487 \cs_new_protected:Npn \DisableHook
\ProvideReversedHook 1488 {
\ProvideMirroredHookPair 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 <@@=
```

\@expl0@@initialize@all@@

```

1524 \cs_new_eq:NN \@expl0@@initialize@all@@
1525   \__hook_initialize_all:
1526 \cs_new_eq:NN \@expl0@@hook@curr@name@pop@@
1527   \__hook_curr_name_pop:

```

(End definition for \@expl0@@initialize@all@@ and \@expl0@@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 `<latexrelease> \NewModuleRelease{2021/06/01}{ltcmdhooks}`
⁵ `<latexrelease> \ExplSyntaxOff` {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
```

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_i: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 `\backslash 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_underscore_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\_scanable:NnTF.)
```

__hook_patch_retokenize:Nnnn Then, if __hook_cmd_if_scanable: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       #####3
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_tmptl
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_tmptl
```

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 <latexrelease>    Generic~hooks~cannot~be~added~to~`#1'.
404  }
405  {
406 <latexrelease>    You~tried~to~add~a~hook~to~`#1',~but~LaTeX~was~unable~to~
407 <latexrelease>    patch~the~command~because~it~\_\_hook\_unpatchable\_cases:n {#2}.
408  }
409 \cs_new:Npn \_\_hook_unpatchable_cases:n #1
410  {
411 <latexrelease>    \str_case:nn {#1}
412  {
413 <latexrelease>      { undef } { doesn't-exist }
414 <latexrelease>      { macro } { is-not-a-macro }
415 <latexrelease>      { expl3 } { is-a-private-expl3-macro }
416 <latexrelease>      { 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.

\@tfor 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
            \@nextwhile = 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
10 \long\def\@whilesw#1\fi{#2\@iwhilesw{#1#2}\fi\fi}
\@iwhilesw
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{%
\@latex@warning@no@line 164     \GenericError{%
\@latex@info 165         \space\space\space\@spaces\@spaces\@spaces
\@latex@info@no@line 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 `tracefont` 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^AT_EX2.09 by \setcounter, \addtocounter or \newcounter for undefined counter. DO NOT use for L^AT_EX2 _{ε} 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^AT_EX2.09 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.¹⁵

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 added¹⁶. 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.

¹⁵Already not quite true: the command `\noindent` starts the paragraph but influences the special processing by suppressing the paragraph indentation box normally inserted by it.

¹⁶That’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. `LATEX` 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

LA**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`.¹⁷

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 thingTM” 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.¹⁸

¹⁷Ideally, `\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.

¹⁸Similarly to the `\everypar` situation, the remedy is that such packages stop doing this and instead add their alterations into the paragraph hooks now provided.

`para/before`
`para/begin`
`para/end`
`para/after`

The following four public hooks are defined and executed for each paragraph:

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.¹⁹

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.²⁰

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.

²⁰Maybe 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
\endgraf
\para_end:
```

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^AT_EX `\endgraf` (with that behavior) was originally also available.

With the new paragraph handling in L^AT_EX, 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^AT_EX'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`²¹ 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_noindent` 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^AT_EX 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^AT_EX 2 _{ε} 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^AT_EX 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.²² 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^AT_EX 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.

²²Note 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 291.)

\@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 292.)

\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^AT_EX 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^AT_EX 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^AT_EX 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 293.)

__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 `\paraomit_indent:` (or `\OmitIndent`) changes that to do nothing.

```

46 \cs_new:Npn \paraomit_indent: {
47   \box_gclear:N \g_para_indent_box
48 }
```

(End definition for `__para_handle_indent:`)

\IndentBox **\OmitIndent** The L^AT_EX2_< 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 \paraomit_indent:
```

(End definition for `\IndentBox` and `\OmitIndent`. These functions are documented on page 293.)

\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_EX 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_EX in certain situations:

- when using `\par` in the code or the document
- when using a blank line (which is converted to `\par`)
- when T_EX finds any commands incompatible with horizontal mode it issues a `\par` and then rereads the command.

Unfortunately, \TeX has some (these days) unnecessary optimizations: if a \vbox ends and \TeX is still in horizontal mode it simply exercises the paragraph builder instead of issuing a \par . It is therefore necessary for \LaTeX to ensure that this case doesn't happen and all boxes internally have a \par command at their end.

This \par may or may not run the “par primitive” (which is always available as \tex_par:D in expl3); it is permissible to have a changed meaning and it is in fact changed by \LaTeX in various ways at various points inside latex.ltx . For this $\text{\LaTeX} 2_{\varepsilon}$ code has the following conventions: \@@par and \endgraf both refer to the default meaning (in the past this was the initex primitive) while \par is the current meaning which maybe does something else.

We are now going to change this default meaning to instead run \para_end: , which ultimately executes the 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 \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 \tex_par:D . This way we don't have to care about whether \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 \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 \if_mode... conditions would be even faster but has the danger to conflict with conditionals in the user hooks.

If \para_end: is executed while \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 \TeX is in hmode while that assignment happens then the \par is seen in hmode because in the above case the assignment may not be finished (one should have used \z@skip) and the \lst@vskip will get inserted into the middle of the conditional. The \lst@vskip then changes to vmode and you get a surprising error about the para/end hook having changed modes even if you don't have any hook code(!): it is the inserted \lst@vskip that is actually causing the change of mode. This is what happened when the output routines got started while a lstlisting environment (that redefines \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 292.*)

`\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 293.*)

```

\RawIndent The LATEX 2 $\varepsilon$  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 293.)

This ends the para module code.

```
97 <@>
```

```

\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 292.)

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

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

```

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^AT_EX supports the plain T_EX 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^AT_EX 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\hspacer\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 = 0pt 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 p

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_{2ε} 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 q

ltfiles.dtx

1 File Handling

The following user commands are defined in this part:

\document	(ie \begin{document})
\nofiles	Reads in the .AUX files and \catcode's @ to 12.
\includeonly	Suppresses all file output by setting \@filesw false. \{NAME1, ... ,NAMEn\}
\include	Causes only parts NAME1, ... ,NAMEn to be read by their \include commands. Works by setting partsw true and setting \@partlist to NAME1, ... ,NAMEn. \{NAME\}
\input	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. \{NAME\}
\IfFileExists	The same as TeX's \input, except it allows optional braces around the file name. In LATEX 2 _E , it also avoids the primitive 'missing file' error, if the file can not be found. \{NAME\}\{then\}\{else\}
\InputIfFileExists	If the file exists on the system, execute then otherwise execute else. \{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 \@parts w = 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^AT_EX 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 \if@filesw and \if@partsw.*)

\@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 \ExplSyntax@load@backend@@
18 \ifx\@unusedoptionlist\empty\else
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 \firstcolumntrue
```

```

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^AT_EX2.0x and plain T_EX.)

```

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\m@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}

```

These internal hooks are already declared earlier (in `ltxexpl`) so that other modules could write to them.

```

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 \if@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@files
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\sfcodes`.=\@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<isp\global\topskip isp\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@filesw
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 \@nodocument \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 `\@fileswfalse` which suppresses the places where L^AT_EX makes `\immediate` writes. The `\makeindex` and `\maketoc` 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 `\if@nobreak` test is needed.

```

198 \def\nofiles{%
199   \@fileswfalse
200   \typeout{No auxiliary output files.^J}%
201   \long\def\protected@write##1##2##3{%
202     {\@write\m@ne{}\if@nobreak\ifvmode\nobreak\fi\fi}%
203   \let\makeindex\relax
204   \let\maketoc\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   \if@nobreak\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^AT_EX 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}٪
```

For historical reasons \@include expects an argument delimited by a space. This is kept
(though unnecessary now) to avoid errors in other packages that use \@include directly.

```
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
\@strip@tex@ext@aux 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}##1%
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>

```

`\@include`

```

290 <|/2ekernel>
291 <|*2ekernel | latexrelease>

```

```

292  \langle latexrelease \rangle \IncludeInRelease{2020/10/01}%
293  \langle latexrelease \rangle
294  \def\@include#1 {%
295  \ifx\@nodocument\relax
296  \clearpage
297  \if@files w
298  \immediate\write\mainaux{\string\@input{#1.aux}}%
299  \fi
300  \if@tempswat rue
301  \if@partsw
302  \if@tempswaf alse
303  \edef\reserved@b{#1}%
304  \for\reserved@a:=\partlist\do
305  {\ifx\reserved@a\reserved@b\@tempswat rue\fi}%
306  \fi
307  \if@tempswa
308  \let\@auxout\partaux
309  \if@files w
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@files w
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 \textrun{EndIncludeInRelease}{  
347 /2ekernel | latexrelease}  
  
348 \textrun{IncludeInRelease[0000/00/00]}{  
349 \textrun{@include}{Spaces in file names}}%  
350 \textrun{def\@include\#1}{%  
351 \textrun{clearpage}{  
352 \textrun{if@filesw}{  
353 \textrun{immediate\write\mainaux{\string\input{\#1.aux}}}{  
354 \textrun{fi}{  
355 \textrun{@tempswattrue}{  
356 \textrun{if@partsw}{  
357 \textrun{@tempswafalse}{  
358 \textrun{edef\reserved@b{\#1}}}{  
359 \textrun{@for\reserved@a:=\partlist\do}{  
360 \textrun{\ifx\reserved@a\reserved@b\@tempswattrue\fi}}}{  
361 \textrun{fi}{  
362 \textrun{if@tempswa}{  
363 \textrun{let\auxout\partaux}{  
364 \textrun{if@filesw}{  
365 \textrun{immediate\openout\partaux #1.aux}{  
366 \textrun{immediate\write\partaux{\relax}}}{  
367 \textrun{fi}{  
368 \textrun{@input{\#1.tex}}}{  
369 \textrun{clearpage}{  
370 \textrun{@writeckpt{\#1}}}{  
371 \textrun{if@filesw}{  
372 \textrun{immediate\closeout\partaux}{  
373 \textrun{fi}{  
374 \textrun{else}{  
375 \textrun{deadcycles\z@}{  
376 \textrun{@nameuse{cp@\#1}}}{  
377 \textrun{fi}{  
378 \textrun{let\auxout\mainaux}{  
379 \textrun{}}}{  
380 \textrun{EndIncludeInRelease}{  
381 /*2ekernel}}
```

(End definition for `\@include`.)

```

\@writeckpt
 382 \def\@writeckpt#1{%
 383   \if@filesw
 384     \immediate\write\@partaux{\string\@setckpt{#1}\@charlb}%
 385     {\let\@elt\@wckptelt \c@l@ckpt}%
 386     \immediate\write\@partaux{\@charrb}%
 387   \fi}
 388 
 389   (End definition for \@writeckpt.)
```

```

\@wckptelt
 388 \def\@wckptelt#1{%
 389   \immediate\write\@partaux{%
 390     \string\setcounter{#1}{\the\@nameuse{c@#1}}}}
 391 
 392   (End definition for \@wckptelt.)
```

```

\@setckpt RmS 93/08/31: introduced \@setckpt
 391 \def\@setckpt#1{\global\@namedef{cp@#1}}
 392 
 393   (End definition for \@setckpt.)
```

\@charlb The following defines \@charlb and \@charrb to be { and }, respectively with \catcode 11.

```

 392 {\catcode`[=1 \catcode`=2
 393 \catcode`[=11 \catcode`]=11
 394 \gdef\@charlb[{}]
 395 \gdef\@charrb[]}
 396 }% }brace matching
 397 
 398   (End definition for \@charlb and \@charrb.)
```

1.1 Safe Input Macros

\@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\expandafter{%
403      \expandafter\string
404      \csname\@firstofone#1\@empty\endcsname}%
405  \endgroup
406 }

(End definition for \curr@file and \setcurr@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"#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   \setcurr@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{\@iffileonpath{#1}}%
454     \fi
455   \else
456     \closein\@inputcheck
457     \edef\@filef@nd{"#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 <{latexrelease}> \openin\@inputcheck"#1" %
470 <{latexrelease}> \ifeof\@inputcheck
471 <{latexrelease}> \ifx\input@path\@undefined
472 <{latexrelease}>   \def\reserved@a{#3}%
473 <{latexrelease}> \else
474 <{latexrelease}>   \def\reserved@a{\@iffileonpath{#1}{#2}{#3}}%
475 <{latexrelease}> \fi
476 <{latexrelease}> \else
477 <{latexrelease}>   \closein\@inputcheck
478 <{latexrelease}>   \edef\@filef@nd{"#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\@addtolist{#1}\@@input}}}}

```

(End definition for `\InputIfFileExists`.)

`\@swaptwoargs` Swap two arguments and return them unbraced (like `\@firstoftwo` etc).

```

532 {/2ekernel}
533 {*2ekernel | latexrelease}
534 {latexrelease}\IncludeInRelease{2019/10/01}%
535 {latexrelease} {\@swaptwoargs}{Don't lose the file name}%
536 \long\def\@swaptwoargs#1#2{#2#1}

537 {/2ekernel | latexrelease}
538 {latexrelease}\EndIncludeInRelease
539 {latexrelease}\IncludeInRelease{0000/00/00}%
540 {latexrelease} {\@swaptwoargs}{Don't lose the file name}%
541 {latexrelease}\let\@swaptwoargs\@undefined
542 {latexrelease}\EndIncludeInRelease
543 {*2ekernel}

```

(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 \LaTeX).

```
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 {/2ekernel}
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}}%

```

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 {latexrelease}\EndIncludeInRelease

```

```

561 〈latexrelease〉\IncludeInRelease{2019/10/01}%
562 〈latexrelease〉          {\@input}{Quote file names}%
563 〈latexrelease〉
564 〈latexrelease〉\def\@input#1{%
565 〈latexrelease〉  \InputIfFileExists{#1}{}%
566 〈latexrelease〉  {\filename@parse@\curr@file
567 〈latexrelease〉  \edef\reserved@a{\noexpand\@missingfileerror
568 〈latexrelease〉  {\filename@area\filename@base}%
569 〈latexrelease〉  {\ifx\filename@ext\relax tex\else\filename@ext\fi}%
570 〈latexrelease〉  \reserved@a}%
571 〈latexrelease〉\EndIncludeInRelease
572 〈latexrelease〉\IncludeInRelease{0000/00/00}%
573 〈latexrelease〉          {\@input}{Quote file names}%
574 〈latexrelease〉\def\@input#1{%
575 〈latexrelease〉  \InputIfFileExists{#1}{}%
576 〈latexrelease〉  {\filename@parse{#1}%
577 〈latexrelease〉  \edef\reserved@a{\noexpand\@missingfileerror
578 〈latexrelease〉  {\filename@area\filename@base}%
579 〈latexrelease〉  {\ifx\filename@ext\relax tex\else\filename@ext\fi}%
580 〈latexrelease〉  \reserved@a}%
581 〈latexrelease〉\EndIncludeInRelease
582 (*2ekernel)

```

(End definition for `\@input`.)

`\@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`
`(*2ekernel | latexrelease)`
`588 〈latexrelease〉\IncludeInRelease{2020/10/01}%`
`589 〈latexrelease〉 {\@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  (latexrelease)          \ifx\filename@ext\relax#2\else\filename@ext\fi}%
638  (latexrelease)      \edef\reserved@a{%
639  (latexrelease)          \noexpand\InputIfFileExists
640  (latexrelease)          {\filename@area\filename@base.\filename@ext}%
641  (latexrelease)          {}}%

```

```

642 〈\latexrelease〉      {\noexpand\@missingfileerror
643 〈\latexrelease〉          {\filename@area\filename@base}{\filename@ext}}}}%
644 〈\latexrelease〉      \reserved@a
645 〈\latexrelease〉      \fi}
646 〈\latexrelease〉
647 〈\latexrelease〉\EndIncludeInRelease
648 {*ekernel}

(End definition for \@missingfileerror.)

```

\@obsoletefile For compatibility with L^AT_EX 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@@{%
657     \def\reserved@d{\\"}%
658     \atfor\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 tex\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 \only\listfiles

```
\@dofilelist 678 \let\@dofilelist\relax
679 \end{2ekernel}
```

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

File r

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^AT_EX 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^AT_EX 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@\ifdefinable\@rc@\ifdefinable
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 `\ifdefinable` (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@\ifdefinable\@rc@\ifdefinable
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 
```

/2ekernel | latexrelease)

- 13 <latexrelease>\EndIncludeInRelease
- 14 <latexrelease>\IncludeInRelease{0000/00/00}{\DeclareTextCompositeCommand}
- 15 <latexrelease> {test for undeclared accent}%
- 16 <latexrelease>\def\DeclareTextCompositeCommand#1#2#3#4{%
- 17 <latexrelease> \expandafter\let\expandafter\reserved@a
- 18 <latexrelease> \csname#2\string#1\endcsname
- 19 <latexrelease> \expandafter\expandafter\expandafter\ifx
- 20 <latexrelease> \expandafter@\car\reserved@a\relax\relax@nil
- 21 <latexrelease> \expandafter\@text@composite \else
- 22 <latexrelease> \edef\reserved@b##1{%
- 23 <latexrelease> \def\expandafter\noexpand
- 24 <latexrelease> \csname#2\string#1\endcsname####1{%
- 25 <latexrelease> \noexpand\@text@composite
- 26 <latexrelease> \expandafter\noexpand\csname#2\string#1\endcsname
- 27 <latexrelease> ####1\noexpand\empty\noexpand\@text@composite
- 28 <latexrelease> {##1}}}%
- 29 <latexrelease> \expandafter\reserved@c\expandafter{\reserved@a{##1}}%
- 30 <latexrelease> \fi
- 31 <latexrelease> \expandafter\def\csname\expandafter\string\csname
- 32 <latexrelease> #2\endcsname\string#1-\string#3\empty\endcsname{#4}}%
- 33 <latexrelease>\EndIncludeInRelease
- 34

*2ekernel>

125 \onlypreamble\DeclareTextCompositeCommand

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 `\\"{}T1\'{} - \string \endcsname`, 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}{\textsuperscript{TM}}
304 \DeclareTextCommandDefault{\SS}{\SS}

305 \DeclareTextCommandDefault{\textordfeminine}{\textsuperscript{a}}
306 \DeclareTextCommandDefault{\textordmasculine}{\textsuperscript{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}{\textcommabove}{comma above}%
354  {*2ekernel | latexrelease}
355  \DeclareTextCommandDefault{\textcommabove}[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}{\textcommabove}{comma above}%
370  \let\textcommabove\@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^AT_EX 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_EX 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 ‘ \TeX math italic’ (OML) encoding. Even though this is meant to be a math font, it includes some of the standard \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 ‘ \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 \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^AT_EX L^IC_R 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#1{%
996     \def\@remove@tlig##1{\@remove@tlig@##1\@nil#1\@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      \newluafunction\@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      token.set_lua('@remove@tlig@@@', rawchar_func, 'global', 'protected')
1029  }}

```

Now `\remove@tlig` can be implemented almost as in XeTeX.

```

1030      \def\remove@tlig#1{\@remove@tlig@@@#1\relax}
1031      \fi
1032  \fi
1033 \else
XeTeX
1034  \def\UnicodeFontTeXLigatures[mapping=tex-text;]{}
1035  \def\remove@tlig#1{\XeTeXglyph\numexpr\XeTeXcharglyph#1\relax}
1036 \fi
1037 \def\UnicodeFontFile#1#2{"[#1]:#2"}
1038 \def\UnicodeFontName#1#2{"#1:#2"}

    Declare the encoding
1039 \DeclareFontEncoding\UnicodeEncodingName{}{}

    Declare accent command to use a postpended combining character rather than the
TeX \accent primitive
1040 \def\add@unicode@accent#1#2{%
1041     \if\relax\detokenize{#2}\relax^\^a0\else#2\fi
1042     \char#1\relax}

```

```

1043 \def\DeclareUnicodeAccent#1#2#3{%
1044   \DeclareTextCommand{#1}{#2}{\add@unicode@accent{#3}}%
1045 }

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.

1046 {
1047   \catcode`z@=11\relax
1048   \gdef\DeclareUnicodeComposite#1#2#3{%
1049     \def\reserved@a##1##2{%
1050       \DeclareTextCompositeCommand#1\UnicodeEncodingName{#2}{%
1051         \iffontchar\font#3 ##2%
1052           \else ##1\fi}%
1053         \expandafter\expandafter\expandafter\extract@default@composite
1054         \csname\UnicodeEncodingName\string#1\endcsname{#2}\@nil
1055       \bgroup
1056         \lccode`z@#3 %
1057         \lowercase{\egroup
1058         \expandafter\reserved@a\expandafter{\reserved@b}{``@}}}}%
1059 }

1060 \def\extract@default@composite#1{%
1061   \ifx\@text@composite#1%
1062     \expandafter\extract@default@composite@a
1063   \else
1064     \expandafter\extract@default@composite@b\expandafter#1%
1065   \fi}
1066 \def\extract@default@composite@a#1\@text@composite#2\@nil{%
1067   \def\reserved@b{#2}}
1068 \def\extract@default@composite@b#1#2\@nil{%
1069   \def\reserved@b{#1#2}}
1070 \DeclareTextCommand{textquotesingle }\UnicodeEncodingName{%
1071   \remove@tlig{"0027}}
1072 \DeclareTextCommand{textasciigrave }\UnicodeEncodingName{%
1073   \remove@tlig{"0060}}
1074 \DeclareTextCommand{textquotedbl }\UnicodeEncodingName{%
1075   \remove@tlig{"0022}}
1076 \DeclareTextSymbol{\textdollar} \UnicodeEncodingName{"0024}
1077 \DeclareTextSymbol{\textless} \UnicodeEncodingName{"003C}
1078 \DeclareTextSymbol{\textgreater} \UnicodeEncodingName{"003E}
1079 \DeclareTextSymbol{\textbackslash} \UnicodeEncodingName{"005C}
1080 \DeclareTextSymbol{\textasciicircum} \UnicodeEncodingName{"005E}
1081 \DeclareTextSymbol{\textunderscore} \UnicodeEncodingName{"005F}
1082 \DeclareTextSymbol{\textbraceleft} \UnicodeEncodingName{"007B}
1083 \DeclareTextSymbol{\textbar} \UnicodeEncodingName{"007C}
1084 \DeclareTextSymbol{\textbraceright} \UnicodeEncodingName{"007D}
1085 \DeclareTextSymbol{\textasciitilde} \UnicodeEncodingName{"007E}
1086 \DeclareTextSymbol{\textexclamdown} \UnicodeEncodingName{"00A1}
1087 \DeclareTextSymbol{\textcent} \UnicodeEncodingName{"00A2}
1088 \DeclareTextSymbol{\textsterling} \UnicodeEncodingName{"00A3}
1089 \DeclareTextSymbol{\textcurrency} \UnicodeEncodingName{"00A4}
1090 \DeclareTextSymbol{\textyen} \UnicodeEncodingName{"00A5}

```

```

1091 \DeclareTextSymbol{\textbrokenbar}           \UnicodeEncodingName{"00A6}
1092 \DeclareTextSymbol{\textsection}             \UnicodeEncodingName{"00A7}
1093 \DeclareTextSymbol{\textasciidieresis}        \UnicodeEncodingName{"00A8}
1094 \DeclareTextSymbol{\textcopyright}            \UnicodeEncodingName{"00A9}
1095 \DeclareTextSymbol{\textordfeminine}          \UnicodeEncodingName{"00AA}

1096 \DeclareTextSymbol{\guillemetleft}          \UnicodeEncodingName{"00AB}
1097 % old Adobe name
1098 \DeclareTextSymbol{\guillemotleft}           \UnicodeEncodingName{"00AB}
1099 \DeclareTextSymbol{\textlnot}                 \UnicodeEncodingName{"00AC}
1100 \DeclareTextSymbol{\textregistered}           \UnicodeEncodingName{"00AE}
1101 \DeclareTextSymbol{\textascimacron}            \UnicodeEncodingName{"00AF}
1102 \DeclareTextSymbol{\textdegree}                \UnicodeEncodingName{"00B0}
1103 \DeclareTextSymbol{\textppm}                  \UnicodeEncodingName{"00B1}
1104 \DeclareTextSymbol{\texttwosuperior}          \UnicodeEncodingName{"00B2}
1105 \DeclareTextSymbol{\textthreesuperior}         \UnicodeEncodingName{"00B3}
1106 \DeclareTextSymbol{\textasciacute}             \UnicodeEncodingName{"00B4}
1107 \DeclareTextSymbol{\textmu}                   \UnicodeEncodingName{"00B5}
1108 \DeclareTextSymbol{\textparagraph}             \UnicodeEncodingName{"00B6}
1109 \DeclareTextSymbol{\textperiodcentered}         \UnicodeEncodingName{"00B7}
1110 \DeclareTextSymbol{\textonesuperior}            \UnicodeEncodingName{"00B9}
1111 \DeclareTextSymbol{\textordmasculine}          \UnicodeEncodingName{"00BA}

1112 \DeclareTextSymbol{\guillemetright}           \UnicodeEncodingName{"00BB}
1113 % old Adobe name
1114 \DeclareTextSymbol{\guillemotright}            \UnicodeEncodingName{"00BB}
1115 \DeclareTextSymbol{\textonequarter}             \UnicodeEncodingName{"00BC}
1116 \DeclareTextSymbol{\textonehalf}                \UnicodeEncodingName{"00BD}
1117 \DeclareTextSymbol{\textthreequarters}          \UnicodeEncodingName{"00BE}
1118 \DeclareTextSymbol{\textquestiondown}           \UnicodeEncodingName{"00BF}
1119 \DeclareTextSymbol{\textAE}                    \UnicodeEncodingName{"00C6}
1120 \DeclareTextSymbol{\textDH}                    \UnicodeEncodingName{"00D0}
1121 \DeclareTextSymbol{\textttimes}                 \UnicodeEncodingName{"00D7}
1122 \DeclareTextSymbol{\textO}                     \UnicodeEncodingName{"00D8}
1123 \DeclareTextSymbol{\textTH}                   \UnicodeEncodingName{"00DE}
1124 \DeclareTextSymbol{\textss}                   \UnicodeEncodingName{"00DF}
1125 \DeclareTextSymbol{\textae}                   \UnicodeEncodingName{"00E6}
1126 \DeclareTextSymbol{\textdh}                   \UnicodeEncodingName{"00F0}
1127 \DeclareTextSymbol{\textdiv}                  \UnicodeEncodingName{"00F7}
1128 \DeclareTextSymbol{\texto}                   \UnicodeEncodingName{"00F8}
1129 \DeclareTextSymbol{\textth}                  \UnicodeEncodingName{"00FE}
1130 \DeclareTextSymbol{\textDJ}                  \UnicodeEncodingName{"0110}
1131 \DeclareTextSymbol{\textdj}                  \UnicodeEncodingName{"0111}
1132 \DeclareTextSymbol{\texti}                   \UnicodeEncodingName{"0131}
1133 \DeclareTextSymbol{\textIJ}                  \UnicodeEncodingName{"0132}
1134 \DeclareTextSymbol{\textij}                  \UnicodeEncodingName{"0133}
1135 \DeclareTextSymbol{\textL}                   \UnicodeEncodingName{"0141}
1136 \DeclareTextSymbol{\textl}                   \UnicodeEncodingName{"0142}
1137 \DeclareTextSymbol{\textNG}                  \UnicodeEncodingName{"014A}
1138 \DeclareTextSymbol{\textng}                  \UnicodeEncodingName{"014B}
1139 \DeclareTextSymbol{\textOE}                  \UnicodeEncodingName{"0152}
1140 \DeclareTextSymbol{\textoe}                  \UnicodeEncodingName{"0153}
1141 \DeclareTextSymbol{\textflorin}              \UnicodeEncodingName{"0192}
1142 \DeclareTextSymbol{\textj}                   \UnicodeEncodingName{"0237}

```

```

1143 \DeclareTextSymbol{\textasciicaron}          \UnicodeEncodingName{"02C7}
1144 \DeclareTextSymbol{\textasciibreve}          \UnicodeEncodingName{"02D8}
1145 \DeclareTextSymbol{\textacutedbl}            \UnicodeEncodingName{"02DD}
1146 \DeclareTextSymbol{\textgravedbl}            \UnicodeEncodingName{"02F5}
1147 \DeclareTextSymbol{\texttildelow}            \UnicodeEncodingName{"02F7}
1148 \DeclareTextSymbol{\textbaht}                \UnicodeEncodingName{"0E3F}
1149 \DeclareTextSymbol{\SS}                     \UnicodeEncodingName{"1E9E}
1150 \DeclareTextSymbol{\textcompwordmark}        \UnicodeEncodingName{"200C}

1151 \%\\DeclareTextSymbol{\textnonbreakinghyphen} \UnicodeEncodingName{"2011}
1152 \%\\DeclareTextSymbol{\textfiguredash}       \UnicodeEncodingName{"2012}
1153 \DeclareTextSymbol{\textendash}              \UnicodeEncodingName{"2013}
1154 \DeclareTextSymbol{\textemdash}               \UnicodeEncodingName{"2014}
1155 \%\\DeclareTextSymbol{\texthorizontalbar}   \UnicodeEncodingName{"2015}

```

Unfortunately some fonts do not implement "2011, "2012 and/or "2015 (including the L^AT_EX default fonts for Unicode engines) so we provide some approximations if the glyph is missing, like we do for OT1 and T1.

The `\nobreak\hskip\z@` is there to prevent a break after the hyphen but allow later breaks in the remainder of the word.

```

1156 \DeclareTextCommand{\textnonbreakinghyphen} \UnicodeEncodingName
1157   {\liffontchar\font "2011 \char "2011 \else \mbox{-}\nobreak\hskip\z@\fi}
1158 \DeclareTextCommand{\textfiguredash}        \UnicodeEncodingName
1159   {\liffontchar\font "2012 \char "2012 \else \char "2013 \fi}
1160 \DeclareTextCommand{\texthorizontalbar}    \UnicodeEncodingName
1161   {\liffontchar\font "2015 \char "2015 \else \char "2014 \fi}

1162 \DeclareTextSymbol{\textbardbl}           \UnicodeEncodingName{"2016}
1163 \DeclareTextSymbol{\textquotleft}         \UnicodeEncodingName{"2018}
1164 \DeclareTextSymbol{\textquotright}        \UnicodeEncodingName{"2019}
1165 \DeclareTextSymbol{\quotesinglbase}       \UnicodeEncodingName{"201A}
1166 \DeclareTextSymbol{\textquotedblleft}     \UnicodeEncodingName{"201C}
1167 \DeclareTextSymbol{\textquotedblright}    \UnicodeEncodingName{"201D}
1168 \DeclareTextSymbol{\quotedblbase}        \UnicodeEncodingName{"201E}
1169 \DeclareTextSymbol{\textdagger}           \UnicodeEncodingName{"2020}
1170 \DeclareTextSymbol{\textdaggerdbl}        \UnicodeEncodingName{"2021}
1171 \DeclareTextSymbol{\textbullet}           \UnicodeEncodingName{"2022}
1172 \DeclareTextSymbol{\textellipsis}        \UnicodeEncodingName{"2026}
1173 \DeclareTextSymbol{\textperthousand}     \UnicodeEncodingName{"2030}
1174 \DeclareTextSymbol{\textpertenthousand}   \UnicodeEncodingName{"2031}
1175 \DeclareTextSymbol{\guilsinglleft}       \UnicodeEncodingName{"2039}
1176 \DeclareTextSymbol{\guilsinglright}      \UnicodeEncodingName{"203A}
1177 \DeclareTextSymbol{\textreferencemark}    \UnicodeEncodingName{"203B}
1178 \DeclareTextSymbol{\textinterrobang}      \UnicodeEncodingName{"203D}
1179 \DeclareTextSymbol{\textfractionsolidus}   \UnicodeEncodingName{"2044}
1180 \DeclareTextSymbol{\textlquill}           \UnicodeEncodingName{"2045}
1181 \DeclareTextSymbol{\textrquill}          \UnicodeEncodingName{"2046}
1182 \DeclareTextSymbol{\textdiscount}        \UnicodeEncodingName{"2052}
1183 \DeclareTextSymbol{\textcolonmonetary}    \UnicodeEncodingName{"20A1}
1184 \DeclareTextSymbol{\textlira}             \UnicodeEncodingName{"20A4}
1185 \DeclareTextSymbol{\textnaira}            \UnicodeEncodingName{"20A6}
1186 \DeclareTextSymbol{\textwon}              \UnicodeEncodingName{"20A9}
1187 \DeclareTextSymbol{\textdong}             \UnicodeEncodingName{"20AB}
1188 \DeclareTextSymbol{\texteuro}              \UnicodeEncodingName{"20AC}
1189 \DeclareTextSymbol{\textpeso}             \UnicodeEncodingName{"20B1}

```

```

1190 \DeclareTextSymbol{\textcelsius}          \UnicodeEncodingName{"2103}
1191 \DeclareTextSymbol{\textnumero}           \UnicodeEncodingName{"2116}
1192 \DeclareTextSymbol{\textcircledP}          \UnicodeEncodingName{"2117}
1193 \DeclareTextSymbol{\textrecip}             \UnicodeEncodingName{"211E}
1194 \DeclareTextSymbol{\textservicemark}        \UnicodeEncodingName{"2120}
1195 \DeclareTextSymbol{\texttrademark}          \UnicodeEncodingName{"2122}
1196 \DeclareTextSymbol{\textohm}                \UnicodeEncodingName{"2126}
1197 \DeclareTextSymbol{\textmhohm}              \UnicodeEncodingName{"2127}
1198 \DeclareTextSymbol{\textestimated}          \UnicodeEncodingName{"212E}
1199 \DeclareTextSymbol{\textleftarrow}            \UnicodeEncodingName{"2190}
1200 \DeclareTextSymbol{\textuparrow}              \UnicodeEncodingName{"2191}
1201 \DeclareTextSymbol{\textrightarrow}          \UnicodeEncodingName{"2192}
1202 \DeclareTextSymbol{\textdownarrow}            \UnicodeEncodingName{"2193}
1203 \DeclareTextSymbol{\textminus}               \UnicodeEncodingName{"2212}
1204

1205 \DeclareTextSymbol{\Hwithstroke}            \UnicodeEncodingName{"0126}
1206 \DeclareTextSymbol{\hwithstroke}             \UnicodeEncodingName{"0127"}

```

Not all fonts have U+2217 but using U+002A requires some adjustment.

```

1207 \DeclareTextCommand{\textasteriskcentered}\UnicodeEncodingName{%
1208   \iffontchar\font"2217 \char"2217 \else
1209     \begingroup
1210       \fontsize
1211         {\the\dimexpr1.3\dimexpr\f@size pt\relax}%
1212         {\f@baselineskip}%
1213       \selectfont
1214       \raisebox{-0.7ex}{[\dimexpr\height-0.7ex] [0pt]{*}}%
1215     \endgroup
1216   \fi
1217 }

1218 \DeclareTextSymbol{\textsurd}                \UnicodeEncodingName{"221A}
1219 \DeclareTextSymbol{\textangle}                 \UnicodeEncodingName{"2329}
1220 \DeclareTextSymbol{\textrangle}                \UnicodeEncodingName{"232A}
1221 \DeclareTextSymbol{\textblank}                 \UnicodeEncodingName{"2422}
1222 \DeclareTextSymbol{\textvisible}               \UnicodeEncodingName{"2423}
1223 \DeclareTextSymbol{\textopenbullet}             \UnicodeEncodingName{"25E6}
1224 \DeclareTextSymbol{\textbigcircle}              \UnicodeEncodingName{"25EF}
1225 \DeclareTextSymbol{\textmusicalnote}            \UnicodeEncodingName{"266A}
1226 \DeclareTextSymbol{\textmarried}                \UnicodeEncodingName{"26AD}
1227 \DeclareTextSymbol{\textdivorced}                \UnicodeEncodingName{"26AE}
1228 \DeclareTextSymbol{\textinterrobangdown}          \UnicodeEncodingName{"2E18}

```

Accents must be declared before the composites that use them.

```

1229 \DeclareUnicodeAccent{\`{}}                  \UnicodeEncodingName{"0300}
1230 \DeclareUnicodeAccent{\'}                   \UnicodeEncodingName{"0301}
1231 \DeclareUnicodeAccent{\^{\prime}}            \UnicodeEncodingName{"0302}
1232 \DeclareUnicodeAccent{\~{\prime}}            \UnicodeEncodingName{"0303}
1233 \DeclareUnicodeAccent{\=\prime}              \UnicodeEncodingName{"0304}
1234 \DeclareUnicodeAccent{\u{\prime}}            \UnicodeEncodingName{"0306}
1235 \DeclareUnicodeAccent{\.\prime}              \UnicodeEncodingName{"0307}
1236 \DeclareUnicodeAccent{\\"{\prime}}            \UnicodeEncodingName{"0308}
1237 \DeclareUnicodeAccent{\r{\prime}}              \UnicodeEncodingName{"030A}
1238 \DeclareUnicodeAccent{\H{\prime}}              \UnicodeEncodingName{"030B}

```

```

1239 \DeclareUnicodeAccent{\v} \UnicodeEncodingName{"030C}
1240 \DeclareUnicodeAccent{\b} \UnicodeEncodingName{"0332}
1241 \DeclareUnicodeAccent{\d} \UnicodeEncodingName{"0323}
1242 \DeclareUnicodeAccent{\c} \UnicodeEncodingName{"0327}
1243 \DeclareUnicodeAccent{\k} \UnicodeEncodingName{"0328}
1244 \DeclareTextCommand{textcommabelow}{\hmode@bgroup\oalign{\null#1\crcr\hidewidth\raise-.31ex
1245   \hbox{\check@mathfonts\fontsize\ssf@size\z@%
1246     \math@fontsfalse\selectfont,\}\hidewidth}\egroup}
1247
1248 \DeclareUnicodeComposite{\^} {}{"005E}
1249 \DeclareUnicodeComposite{\~} {}{"007E}
1250 \DeclareUnicodeComposite{\'} {\A}{"00C0}
1251 \DeclareUnicodeComposite{\'} {\A}{"00C1}
1252 \DeclareUnicodeComposite{\^} {\A}{"00C2}
1253 \DeclareUnicodeComposite{\~} {\A}{"00C3}
1254 \DeclareUnicodeComposite{\"} {\A}{"00C4}
1255 \DeclareUnicodeComposite{\r} {\A}{"00C5}
1256 \DeclareUnicodeComposite{\c} {\C}{"00C7}
1257 \DeclareUnicodeComposite{\'} {\E}{"00C8}
1258 \DeclareUnicodeComposite{\'} {\E}{"00C9}
1259 \DeclareUnicodeComposite{\^} {\E}{"00CA}
1260 \DeclareUnicodeComposite{\"} {\E}{"00CB}
1261 \DeclareUnicodeComposite{\'} {\I}{"00CC}
1262 \DeclareUnicodeComposite{\'} {\I}{"00CD}
1263 \DeclareUnicodeComposite{\^} {\I}{"00CE}
1264 \DeclareUnicodeComposite{\"} {\I}{"00CF}
1265 \DeclareUnicodeComposite{\~} {\N}{"00D1}
1266 \DeclareUnicodeComposite{\'} {\O}{"00D2}
1267 \DeclareUnicodeComposite{\'} {\O}{"00D3}
1268 \DeclareUnicodeComposite{\^} {\O}{"00D4}
1269 \DeclareUnicodeComposite{\~} {\O}{"00D5}
1270 \DeclareUnicodeComposite{\"} {\O}{"00D6}
1271 \DeclareUnicodeComposite{\'} {\U}{"00D9}
1272 \DeclareUnicodeComposite{\'} {\U}{"00DA}
1273 \DeclareUnicodeComposite{\^} {\U}{"00DB}
1274 \DeclareUnicodeComposite{\"} {\U}{"00DC}
1275 \DeclareUnicodeComposite{\'} {\Y}{"00DD}
1276 \DeclareUnicodeComposite{\'} {\fa}{"00E0}
1277 \DeclareUnicodeComposite{\'} {\fa}{"00E1}
1278 \DeclareUnicodeComposite{\^} {\fa}{"00E2}
1279 \DeclareUnicodeComposite{\~} {\fa}{"00E3}
1280 \DeclareUnicodeComposite{\"} {\fa}{"00E4}
1281 \DeclareUnicodeComposite{\r} {\fa}{"00E5}
1282 \DeclareUnicodeComposite{\c} {\c}{"00E7}
1283 \DeclareUnicodeComposite{\'} {\e}{"00E8}
1284 \DeclareUnicodeComposite{\'} {\e}{"00E9}
1285 \DeclareUnicodeComposite{\^} {\e}{"00EA}
1286 \DeclareUnicodeComposite{\"} {\e}{"00EB}
1287 \DeclareUnicodeComposite{\'} {\i} {"00EC}
1288 \DeclareUnicodeComposite{\'} {\i} {"00ED}
1289 \DeclareUnicodeComposite{\'} {\i} {"00ED}
1290 \DeclareUnicodeComposite{\^} {\i} {"00EE}
1291 \DeclareUnicodeComposite{\~} {\i} {"00EE}

```

```

1292 \DeclareUnicodeComposite{\^}{i}{"00EE}
1293 \DeclareUnicodeComposite{\"}{\i {"00EF}}
1294 \DeclareUnicodeComposite{\~}{i}{"00EF}
1295 \DeclareUnicodeComposite{\~}{n}{"00F1}
1296 \DeclareUnicodeComposite{\'}{o}{"00F2}
1297 \DeclareUnicodeComposite{\'}{o}{"00F3}
1298 \DeclareUnicodeComposite{\^}{o}{"00F4}
1299 \DeclareUnicodeComposite{\~}{o}{"00F5}
1300 \DeclareUnicodeComposite{\"}{o}{"00F6}
1301 \DeclareUnicodeComposite{\'}{u}{"00F9}
1302 \DeclareUnicodeComposite{\'}{\u {"00FA}}
1303 \DeclareUnicodeComposite{\^}{u}{"00FB}
1304 \DeclareUnicodeComposite{\~}{u}{"00FC}
1305 \DeclareUnicodeComposite{\~}{y}{"00FD}
1306 \DeclareUnicodeComposite{\"}{y}{"00FF}
1307 \DeclareUnicodeComposite{\=}{A}{"0100}
1308 \DeclareUnicodeComposite{\=}{a}{"0101}
1309 \DeclareUnicodeComposite{\u}{A}{"0102}
1310 \DeclareUnicodeComposite{\u}{a}{"0103}
1311 \DeclareUnicodeComposite{\k}{A}{"0104}
1312 \DeclareUnicodeComposite{\k}{a}{"0105}
1313 \DeclareUnicodeComposite{\'}{C}{"0106}
1314 \DeclareUnicodeComposite{\'}{c}{"0107}
1315 \DeclareUnicodeComposite{\~}{C}{"0108}
1316 \DeclareUnicodeComposite{\~}{c}{"0109}
1317 \DeclareUnicodeComposite{\.}{C}{"010A}
1318 \DeclareUnicodeComposite{\.}{c}{"010B}
1319 \DeclareUnicodeComposite{\v}{C}{"010C}
1320 \DeclareUnicodeComposite{\v}{c}{"010D}
1321 \DeclareUnicodeComposite{\v}{D}{"010E}
1322 \DeclareUnicodeComposite{\v}{d}{"010F}
1323 \DeclareUnicodeComposite{\=}{E}{"0112}
1324 \DeclareUnicodeComposite{\=}{e}{"0113}
1325 \DeclareUnicodeComposite{\u}{E}{"0114}
1326 \DeclareUnicodeComposite{\u}{e}{"0115}
1327 \DeclareUnicodeComposite{\.}{E}{"0116}
1328 \DeclareUnicodeComposite{\.}{e}{"0117}
1329 \DeclareUnicodeComposite{\k}{E}{"0118}
1330 \DeclareUnicodeComposite{\k}{e}{"0119}
1331 \DeclareUnicodeComposite{\v}{E}{"011A}
1332 \DeclareUnicodeComposite{\v}{e}{"011B}
1333 \DeclareUnicodeComposite{\~}{G}{"011C}
1334 \DeclareUnicodeComposite{\~}{g}{"011D}
1335 \DeclareUnicodeComposite{\u}{G}{"011E}
1336 \DeclareUnicodeComposite{\u}{g}{"011F}
1337 \DeclareUnicodeComposite{\.}{G}{"0120}
1338 \DeclareUnicodeComposite{\.}{g}{"0121}
1339 \DeclareUnicodeComposite{\c}{G}{"0122}
1340 \DeclareUnicodeComposite{\c}{g}{"0123}
1341 \DeclareUnicodeComposite{\~}{H}{"0124}
1342 \DeclareUnicodeComposite{\~}{h}{"0125}
1343 \DeclareUnicodeComposite{\~}{I}{"0128}
1344 \DeclareUnicodeComposite{\~}{i {"0129}}
1345 \DeclareUnicodeComposite{\~}{i {"0129}}

```

```

1346 \DeclareUnicodeComposite{\=}{I}{012A}
1347 \DeclareUnicodeComposite{\=}{i}{012B}
1348 \DeclareUnicodeComposite{\=}{i}{012B}
1349 \DeclareUnicodeComposite{\u}{I}{012C}
1350 \DeclareUnicodeComposite{\u}{i}{012D}
1351 \DeclareUnicodeComposite{\u}{i}{012D}
1352 \DeclareUnicodeComposite{\k}{I}{012E}
1353 \DeclareUnicodeComposite{\k}{i}{012F}
1354 \DeclareUnicodeComposite{\k}{i}{012F}
1355 \DeclareUnicodeComposite{\.}{I}{0130}
1356 \DeclareUnicodeComposite{\^}{J}{0134}
1357 \DeclareUnicodeComposite{\^}{j}{0135}
1358 \DeclareUnicodeComposite{\^}{j}{0135}
1359 \DeclareUnicodeComposite{\c}{K}{0136}
1360 \DeclareUnicodeComposite{\c}{k}{0137}
1361 \DeclareUnicodeComposite{\'}{L}{0139}
1362 \DeclareUnicodeComposite{\'}{l}{013A}
1363 \DeclareUnicodeComposite{\c}{l}{013B}
1364 \DeclareUnicodeComposite{\c}{l}{013C}
1365 \DeclareUnicodeComposite{\v}{L}{013D}
1366 \DeclareUnicodeComposite{\v}{l}{013E}
1367 \DeclareUnicodeComposite{\'}{N}{0143}
1368 \DeclareUnicodeComposite{\'}{n}{0144}
1369 \DeclareUnicodeComposite{\c}{N}{0145}
1370 \DeclareUnicodeComposite{\c}{n}{0146}
1371 \DeclareUnicodeComposite{\v}{N}{0147}
1372 \DeclareUnicodeComposite{\v}{n}{0148}
1373 \DeclareUnicodeComposite{\=}{O}{014C}
1374 \DeclareUnicodeComposite{\=}{o}{014D}
1375 \DeclareUnicodeComposite{\u}{O}{014E}
1376 \DeclareUnicodeComposite{\u}{o}{014F}
1377 \DeclareUnicodeComposite{\H}{O}{0150}
1378 \DeclareUnicodeComposite{\H}{o}{0151}
1379 \DeclareUnicodeComposite{\'}{R}{0154}
1380 \DeclareUnicodeComposite{\'}{r}{0155}
1381 \DeclareUnicodeComposite{\c}{R}{0156}
1382 \DeclareUnicodeComposite{\c}{r}{0157}
1383 \DeclareUnicodeComposite{\v}{R}{0158}
1384 \DeclareUnicodeComposite{\v}{r}{0159}
1385 \DeclareUnicodeComposite{\'}{S}{015A}
1386 \DeclareUnicodeComposite{\'}{s}{015B}
1387 \DeclareUnicodeComposite{\^}{S}{015C}
1388 \DeclareUnicodeComposite{\^}{s}{015D}
1389 \DeclareUnicodeComposite{\c}{S}{015E}
1390 \DeclareUnicodeComposite{\c}{s}{015F}
1391 \DeclareUnicodeComposite{\v}{S}{0160}
1392 \DeclareUnicodeComposite{\v}{s}{0161}
1393 \DeclareUnicodeComposite{\c}{T}{0162}
1394 \DeclareUnicodeComposite{\c}{t}{0163}
1395 \DeclareUnicodeComposite{\v}{T}{0164}
1396 \DeclareUnicodeComposite{\v}{t}{0165}
1397 \DeclareUnicodeComposite{\~}{U}{0168}
1398 \DeclareUnicodeComposite{\~}{u}{0169}
1399 \DeclareUnicodeComposite{\=}{U}{016A}

```

```

1400 \DeclareUnicodeComposite{\=} {u}{"016B}
1401 \DeclareUnicodeComposite{\u} {U}{"016C}
1402 \DeclareUnicodeComposite{\u} {u}{"016D}
1403 \DeclareUnicodeComposite{\r} {U}{"016E}
1404 \DeclareUnicodeComposite{\r} {u}{"016F}
1405 \DeclareUnicodeComposite{\H} {U}{"0170}
1406 \DeclareUnicodeComposite{\H} {u}{"0171}
1407 \DeclareUnicodeComposite{\k} {U}{"0172}
1408 \DeclareUnicodeComposite{\k} {u}{"0173}
1409 \DeclareUnicodeComposite{\^} {W}{"0174}
1410 \DeclareUnicodeComposite{\^} {w}{"0175}
1411 \DeclareUnicodeComposite{\^} {Y}{"0176}
1412 \DeclareUnicodeComposite{\^} {y}{"0177}
1413 \DeclareUnicodeComposite{\"} {Y}{"0178}
1414 \DeclareUnicodeComposite{\'} {Z}{"0179}
1415 \DeclareUnicodeComposite{\'} {z}{"017A}
1416 \DeclareUnicodeComposite{\.} {Z}{"017B}
1417 \DeclareUnicodeComposite{\.} {z}{"017C}
1418 \DeclareUnicodeComposite{\v} {Z}{"017D}
1419 \DeclareUnicodeComposite{\v} {z}{"017E}
1420 \DeclareUnicodeComposite{\v} {A}{"01CD}
1421 \DeclareUnicodeComposite{\v} {a}{"01CE}
1422 \DeclareUnicodeComposite{\v} {I}{"01CF}
1423 \DeclareUnicodeComposite{\v} {i} {"01D0}
1424 \DeclareUnicodeComposite{\v} {i} {"01D0}
1425 \DeclareUnicodeComposite{\v} {O} {"01D1}
1426 \DeclareUnicodeComposite{\v} {o} {"01D2}
1427 \DeclareUnicodeComposite{\v} {U} {"01D3}
1428 \DeclareUnicodeComposite{\v} {u} {"01D4}

1429 \DeclareUnicodeComposite{\'} {\AE} {"01FC}
1430 \DeclareUnicodeComposite{\'} {\E} {"01FC}
1431 \DeclareUnicodeComposite{\'} {\ae} {"01FD}
1432 \DeclareUnicodeComposite{\=} {\aa} {"01FD}
1433 \DeclareUnicodeComposite{\=} {\AE} {"01E2}
1434 \DeclareUnicodeComposite{\=} {\E} {"01E2}
1435 \DeclareUnicodeComposite{\=} {\ae} {"01E3}
1436 \DeclareUnicodeComposite{\=} {\aa} {"01E3}
1437 \DeclareUnicodeComposite{\v} {\G} {"01E6}
1438 \DeclareUnicodeComposite{\v} {\g} {"01E7}
1439 \DeclareUnicodeComposite{\v} {\K} {"01E8}
1440 \DeclareUnicodeComposite{\v} {\k} {"01E9}
1441 \DeclareUnicodeComposite{\k} {\O} {"01EA}
1442 \DeclareUnicodeComposite{\k} {\o} {"01EB}
1443 \DeclareUnicodeComposite{\v} {\j} {"01F0}
1444 \DeclareUnicodeComposite{\v} {\j} {"01F0}
1445 \DeclareUnicodeComposite{\'} {\G} {"01F4}
1446 \DeclareUnicodeComposite{\'} {\g} {"01F5}
1447 \DeclareUnicodeComposite{\textcommabelow}{S} {"0218}
1448 \DeclareUnicodeComposite{\textcommabelow}{s} {"0219}
1449 \DeclareUnicodeComposite{\textcommabelow}{T} {"021A}
1450 \DeclareUnicodeComposite{\textcommabelow}{t} {"021B}
1451 \DeclareUnicodeComposite{\=} {\Y} {"0232}
1452 \DeclareUnicodeComposite{\=} {\y} {"0233}

```

```

1453 \DeclareUnicodeComposite{\.}{B}{1E02}
1454 \DeclareUnicodeComposite{\.}{b}{1E03}
1455 \DeclareUnicodeComposite{\d}{B}{1E04}
1456 \DeclareUnicodeComposite{\d}{b}{1E05}
1457 \DeclareUnicodeComposite{\d}{D}{1E0C}
1458 \DeclareUnicodeComposite{\d}{d}{1E0D}
1459 \DeclareUnicodeComposite{\=}{G}{1E20}
1460 \DeclareUnicodeComposite{\=}{g}{1E21}
1461 \DeclareUnicodeComposite{\d}{H}{1E24}
1462 \DeclareUnicodeComposite{\d}{h}{1E25}
1463 \DeclareUnicodeComposite{\d}{K}{1E32}
1464 \DeclareUnicodeComposite{\d}{k}{1E33}
1465 \DeclareUnicodeComposite{\d}{L}{1E36}
1466 \DeclareUnicodeComposite{\d}{l}{1E37}
1467 \DeclareUnicodeComposite{\d}{M}{1E42}
1468 \DeclareUnicodeComposite{\d}{m}{1E43}
1469 \DeclareUnicodeComposite{\d}{N}{1E46}
1470 \DeclareUnicodeComposite{\d}{n}{1E47}
1471 \DeclareUnicodeComposite{\d}{R}{1E5A}
1472 \DeclareUnicodeComposite{\d}{r}{1E5B}
1473 \DeclareUnicodeComposite{\d}{S}{1E62}
1474 \DeclareUnicodeComposite{\d}{s}{1E63}
1475 \DeclareUnicodeComposite{\d}{T}{1E6C}
1476 \DeclareUnicodeComposite{\d}{t}{1E6D}
1477 \DeclareUnicodeComposite{\d}{V}{1E7E}
1478 \DeclareUnicodeComposite{\d}{v}{1E7F}
1479 \DeclareUnicodeComposite{\d}{W}{1E88}
1480 \DeclareUnicodeComposite{\d}{w}{1E89}
1481 \DeclareUnicodeComposite{\d}{Z}{1E92}
1482 \DeclareUnicodeComposite{\d}{z}{1E93}
1483 \DeclareUnicodeComposite{\d}{A}{1EA0}
1484 \DeclareUnicodeComposite{\d}{a}{1EA1}
1485 \DeclareUnicodeComposite{\d}{E}{1EB8}
1486 \DeclareUnicodeComposite{\d}{e}{1EB9}
1487 \DeclareUnicodeComposite{\d}{I}{1ECA}
1488 \DeclareUnicodeComposite{\d}{i}{1ECB}
1489 \DeclareUnicodeComposite{\d}{O}{1ECC}
1490 \DeclareUnicodeComposite{\d}{o}{1ECD}
1491 \DeclareUnicodeComposite{\d}{U}{1EE4}
1492 \DeclareUnicodeComposite{\d}{u}{1EE5}
1493 \DeclareUnicodeComposite{\d}{Y}{1EF4}
1494 \DeclareUnicodeComposite{\d}{y}{1EF5}

1495 </TU>

```

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.

1496 `(*package)`

Here we define a macro that extends the `\@uclclist` if needed and afterwards turns itself in a noop.

```
1497 \def\update@uclc@with@cyrillic{%
1498   \expandafter\def\expandafter@\@uclclist\expandafter
1499   { \@uclclist
1500     \cyla\CYRA\cyrabhch\CYRABHCH\cyrabhchdsc\CYRABHCHDSC\cyrabhdze
1501     \CYRABHDZE\cyrabhha\CYRABHHA\cyrae\CYRAE\cyrb\CYRB\cyrbyus
1502     \CYRBYUS\cycr\CYRC\cyrch\CYRCH\cyrchldsc\CYRCHLDSC\cyrchrds
1503     \CYRCHRDS\cyrchvcrs\CYRCHVCRS\cyrd\CYRD\cyrdelta\CYRDELTA
1504     \cyrdje\CYRDJE\cyrdze\CYRDZE\cyrdzhe\CYRDZHE\cyre\CYRE\cyreps
1505     \CYREPS\cyrerev\CYREREV\cyry\CYRERY\cyrf\CYRF\cyrfita
1506     \CYRFITA\cyrq\CYRG\cyrgdsc\CYRGDSC\cyrgdschcrs\CYRGDSCHCRS
1507     \cyrghcrs\CYRGHCRS\cyrghk\CYRGHK\cyrgup\CYRGUP\cyrh\CYRH
1508     \cyrhdsc\CYRHDSC\cyrhhcrs\CYRHHCRS\cyrhhk\CYRHHK\cyrhrdsn
1509     \CYRHRDSN\ciri\CYRI\cyrie\CYRIE\ciri\CYRII\cyrishrt\CYRISHRT
1510     \cyrishrtsc\CYRISHRTDSC\cyrizh\CYRIZH\cyrje\CYRJE\cyrk\CYRK
1511     \cyrkbeak\CYRKBEAK\cyrkdsc\CYRKDSC\cyrkhcrs\CYRKHCRS\cyrkhk
1512     \CYRKHK\cyrkvcrs\CYRKVCRS\cyl\CYRL\cyrlsc\CYRLDSC\cylrhk
1513     \CYRLHK\cylje\CYRLJE\cyrm\CYRM\cyrm\cyrmdsc\CYRM\cyrmhk\CYRMHK
1514     \cyrn\CYRN\cyrndsc\CYRNDSC\cyrng\CYRNG\cyrnhk\CYRNHK\cyrnje
1515     \CYRNJE\cyrnlhk\CYRNLHK\cyro\CYRO\cyrotld\CYROTLD\cyrp\CYRP
1516     \cyrphk\CYRPHK\cyrq\CYRQ\cyr\CYRR\cyrrdsc\CYRRDSC\cyrhk
1517     \CYRRHK\cyrrt\CYRRTICK\crys\CYRS\crysacrs\CYRSACRS
1518     \cryscha\CYRSCHWA\crysdc\CYRSDSC\cyrsemisftsn\CYRSEMISFTSN
1519     \crysftsn\CYRSFTSN\cyrsh\CYRSH\cyrshch\CYRSHCH\cyrshha\CYRSHHA
1520     \cyrt\CYRT\cyrtsc\CYRTDSC\cyrtsetse\CYRTETSE\cyrtshe\CYRTSHE
1521     \cyr\CYRU\cyrushrt\CYRUSHRT\cyrv\CYRV\cyrw\CYRW\ciry\CYRY
1522     \crysya\CYRYA\crysat\CYRYAT\crys\crys\crys\crys\crys\crys\crys
1523     \CYRYO\crysya\CYRYU\crys\crys\crys\crys\crys\crys\crys\crys
1524     \cyrzhdsc\CYRZHDSC\%
1525   \let\update@uclc@with@cyrillic\relax
1526 }
```

Here we process each option:

```
1527 \DeclareOption*{%
1528   \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).

```
1529   \tempswafalse
1530   \cifl@t@r\fmtversion{2020/02/02}\%
```

```

1531      {\expandafter\ifx\csname T@\CurrentOption\endcsname\relax
1532          \@tempswattrue\fi}%
1533      {\@tempswattrue}%

```

Load if necessary:

```

1534 \if@tempswa
1535     \edef\reserved@f{%
1536         \lowercase{\def\noexpand\reserved@f{\CurrentOption enc.def}}%
1537     \reserved@f
1538     \InputIfFileExists\reserved@f
1539         {}{\PackageError{fontenc}%
1540             {Encoding file '\reserved@f' not found.%%
1541             \MessageBreak
1542             You might have misspelt the name of the encoding}%
1543             {Necessary code for this encoding was not
1544             loaded.\MessageBreak
1545             Thus calling the encoding later on will
1546             produce further error messages.}%
1547     \let\reserved@f\relax

```

In case the current encoding is one of a list of known cyrillic ones we extend the `\@uclclist`:

```

1548     \expandafter\in@\expandafter{\CurrentOption}%
1549             {T2A,T2B,T2C,X2,LCY,OT2}%
1550     \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.

```

1551     \expandafter\in@\expandafter\cyra\expandafter
1552             {\@uclclist}%
1553     \ifin@
1554     \else
1555         \update@uclc@with@cyrillic
1556     \fi
1557     \fi
1558 \fi
1559 }

```

```
1560 \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^AT_EX previously used:

```
1561 \%fontencoding\encodingdefault\selectfont
```

So instead we do this here:

```
1562 \usefont\encodingdefault\familydefault\seriesdefault\shapedefault
```

To save some space we get rid of the macro extending the `\@uclclist` (might have happened already).

```
1563 \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.

```
1564 \let\@elt\relax
1565 \xdef\@fontenc@load@list{\@fontenc@load@list
1566   \@elt{\csname opt@fontenc.sty\endcsname}}
1567 \global\expandafter\let\csname ver@fontenc.sty\endcsname\relax
1568 \global\expandafter\let\csname opt@fontenc.sty\endcsname\relax
1569 \global\let\@ifl@ter@@\@ifl@ter
1570 \def\@ifl@ter#1#2#3#4#5{\global\let\@ifl@ter\@ifl@ter@@}
1571 
```

File s

ltcounts.dtx

1 Counters and Lengths

Commands for defining and using counters. This file defines:

\newcounter	To define a new counter.
\setcounter	To set the value of counters.
\addtocounter	Increase the counter #1 by the number #2.
\stepcounter	Increase a counter by one.
\refstepcounter	Increase a counter by one, also setting the value used by \label.
\value	For accessing the value of the counter as a TeX number (as opposed to \the<counter>) which expands to the <i>printed</i> representation of <counter>
\arabic	\arabic{<counter>}: 1, 2, 3, ...
\roman	\roman{<counter>}: i, ii, iii, ...
\Roman	\Roman{<counter>}: I, II, III, ...
\alph	\alph{<counter>}: a, b, c, ...
\Alph	\Alph{<counter>}: A, B, C, ...
\fnsymbol	\fnsymbol{<counter>}: *, †, ‡, ...
\counterwithin	\counterwithin[<format>]{<counter>}{<within-counter>}: Resets <counter> whenever <within-counter> is stepped. Also redefines \the<counter> command to produce \the<within-counter>.<format>{<counter>} with \arabic as the default for <format>. Star form omits redefining the print representation.
\counterwithout	\counterwithout[<format>]{<counter>}{<within-counter>}: Removes <counter> from the reset list of <within-counter>. Also redefines \the<counter> command to produce <format>{<counter>} with \arabic as the default for <format>. Star form omits redefining the print representation.

1 {*2ekernel}

1.1 Environment Counter Macros

An environment foo has an associated counter defined by the following control sequences:

\c@foo	Contains the counter's numerical value. It is defined by \newcount\foocounter.
\thefoo	Macro that expands to the printed value of \foocounter. For example, if sections are numbered within chapters, and section headings look like
	Section II-3. The Nature of Counters
	then \thesection might be defined by:
	\def\thesection {\@Roman{\c@chapter}-\@arabic{\c@section}}
\p@foo	Macro that expands to a printed 'reference prefix' of counter foo. Any \ref to a value created by counter foo will produce the expansion of \p@foo\thefoo when the \label command is executed. See file ltxref.dtx for an extension of this mechanism.
\cl@foo	List of counters to be reset when foo stepped. Has format \@elt{counterA}\@elt{counterB}\@elt{counterC}.

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\@ifdefinable \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   \@ifundefined{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
       #3}}}

```

(End definition for \@ifbothcounters.)

```

67  </2ekernel | latexrelease>
68  <|latexrelease>\EndIncludeInRelease
69  <|latexrelease>\IncludeInRelease{0000-00-00}
70  <|latexrelease>                                {\@removefromreset}{Add interfaces}%
71  <|latexrelease>\let \@removefromreset \undefined
72  <|latexrelease>\let \@ifbothcounters \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^AT_EX (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} \|\or
170   \TextOrMath{\textasteriskcentered}\textasteriskcentered\{**}\or
171   \TextOrMath{\textdagger}\textdagger\{ \dagger\dagger\}\or
172   \TextOrMath{\textdaggerdbl}\textdaggerdbl\{ \ddagger\ddagger\}\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 \|\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 t

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 345.4666666pt when calculating a dimension.

```
22 \def\@settopoint#1{\divide#1\p@\multiply#1\p@}
23 </2ekernel>
```

(*End definition for \@settopoint.*)

File u

ltfssbas.dtx

This file contains the main implementation of the ‘low level’ font selection commands. See other parts of the L^AT_EX distribution, or *The L^AT_EX Companion* for higher level documentation of the L^AT_EX ‘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#1 invalid 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_EX 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#1 is\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).²³

```
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`.)

²³For 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`²⁴ 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`.)

²⁴We 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^AT_EX 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_EX function must be hidden.

Anyway, since we don't have this and won't have it in L_AT_EX 2_E 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> <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> <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> <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> <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.
`\@defaultsubs`

```

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^AT_EX.

```
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_EX.

```
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^TE_X version. When using system fonts Xe^TE_X 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 v

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^AT_EX distribution, or *The L^AT_EX Companion* for higher level documentation of the L^AT_EX 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}{sbx}{}  

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 {sle}{?m}{sl}{}
366 \DeclareFontSeriesChangeRule {slsc}{?m}{sl}{}
367 \DeclareFontSeriesChangeRule {slsc}{?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
```
- (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 w

## 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 > @ne
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 `every sel` 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
```

```

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 *(dimen)* 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]#2\@nil{%
414 \def\mandatory@arg{#2}%
415 \def\optional@arg{#1}%
416 }/{2ekernel}
```

(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 }/{2ekernel}
```

(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 }/{trace}
429 {-trace} \csname s@fct@#1\endcsname
430 }
431 }/{2ekernel | package}
```

(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}{\sub@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 x

# 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 \@latex@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.

```
109 \gdef\newmathalphabet{%
110 \if@no@font@opt
111 @latex@error{*** NFSS release 1 command
112 \noexpand\newmathalphabet found%
113 ^^J \space*** Automatic recovery not possible.%
114 ^^J \space*** TYPE H for Help%
115 }%
116 {Please look at the file usrguide.tex for hints on
117 how to resolve this problem.}%
118 \else
119 \warn@rel@i\newmathalphabet\DeclareMathAlphabet
120 \fi
121 \@ifstar\newmathalphabet@@
122 \newmathalphabet@@}%
123 \gdef\newmathalphabet@@{\DeclareMathAlphabet#1{U}{\{}{\}}{}}%
124 \gdef\newmathalphabet@@@#1#2#3#4{%
125 \DeclareMathAlphabet{#1}{U}{#2}{#3}{#4}}%
126 \onlypreamble\newmathalphabet
127 \onlypreamble\newmathalphabet@@
128 \onlypreamble\newmathalphabet@@@
```

(*End definition for \newmathalphabet , \newmathalphabet@@ , and \newmathalphabet@@@.*)

\if@no@font@opt  
\@no@font@optfalse

```
129 \global\let\if@no@font@opt\iftrue
130 \gdef\@no@font@optfalse{\let\if@no@font@opt\iffalse}%
```

(*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 y

## 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  $\backslash reserved@e$  (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  $\backslash version@list$ .)

**\version@elt**

```
18 \let\version@elt\relax
19 \onlypreamble\version@elt
```

(End definition for  $\backslash version@elt$ .)

**\new@mathversion** The macro  $\backslash new@mathversion$  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  $\backslash version@list$ . We enclose  $\backslash version@list$  in braces since it might be empty (if no *version* is defined yet). But this means that we need a suitable number of  $\backslash expandafter$  primitives.

```
21 \% \expandafter\in@\expandafter#1\expandafter{\version@list}%
22 \% \ifin@
```

If so it prints an error message. The  $\backslash next$  macro is used to get rid of the four characters  $\backslash mv@$  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  $\backslash version@list$ . This is very easy: we define  $\backslash version@elt$  (which is the delimiter in  $\backslash version@list$ ) to protect itself and the following token from being expanded and simply redefine  $\backslash version@list$ .

```
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 \relax
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 \fontinfo{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 }
208
```

If there was no change, we report that in the log (but this branch could go completely).

```

209 @font@info{No~math~alphabet~change~to~frozen~version~#1}
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 \expandafter\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 \{\ 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 \if@tempswafalse
405 \edef\reserved@b{#2}%
406 \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
407 \ifx\reserved@b\reserved@c \if@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 \else
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
```

Check if it is already a math alphabet.

```

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 \csname M@#2\expandafter\endcsname
544 \csname \expandafter\@gobble`string#1`space\endcsname\}%
545 \version@list
546 \else
```

Check if it is a math alphabet defined via \DeclareSymbolFontAlphabet.

```

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 \else
688 \@latex@error{Math version `expandafter@gobblefour/string#1'
689 is not
690 defined}{You probably misspelled the name of the math
691 version.^JOr you have to specify an additional package.}%
692 \fi
693 }
694 \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 % }
1062 % @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@tempswa

```

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 z

## 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\sfdefault\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`.

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}%

```

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}\let\init@series@setup\relax
121 \begin{macro}{\textrm}
122 \begin{macro}{\textrm}\EndIncludeInRelease
123 \begin{macro}{\textrm}{*2ekernel}
124 \begin{macro}{\textrm}{/2ekernel}
125 \begin{macro}{\textrm}{*2ekernel | \textrm{}}{2ekernel}
126 \begin{macro}{\textrm}{\IncludeInRelease{2020/02/02}}
127 \begin{macro}{\textrm}{\mdseries@rm}{Custom series}
```

`\mdseries@rm` We initialize the family specific default at the end of the format generation. Later on  
`\mdseries@sf` they may get overwritten in the preamble or a package via `\DeclareFontSeriesDefault`  
`\mdseries@tt` (or possibly directly).

`\bfseries@rm` Conceptual change: The `\bfdefault` will be `b` not `bx` because that is what it should  
`\bfseries@sf` be really for nearly every font except Computer/Latin Modern.

`\bfseries@tt` 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  
`\bfseries@tt` then `\bfdefault` will get used.

```
128 \def\bfs@rm{bx}
129 \def\bfs@sf{bx}
130 \def\bfs@tt{bx}
```

Frozen version of the kernel defaults so we can see if they have changed.

```
131 \let\bfs@rm@kernel\bfs@rm
132 \let\bfs@sf@kernel\bfs@sf
133 \let\bfs@tt@kernel\bfs@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 \begin{macro}{*debug}
138 \let\series@change@debug\typeout
139 \let\series@change@debug\@gobble
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|v|`
- 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
bfseries 423 \NewHook{expand@font@defaults}
bfseries/defaults 424 \NewHook{bfseries}
mdseries 425 \NewHook{bfseries/defaults}
mdseries/defaults 426 \NewHook{mdseries}
427 \NewHook{mdseries/defaults}
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 until they aren't any longer used.

By default the hooks do nothing.

```

\@defaultfamilyhook 429 \let\@rmfamilyhook\@empty
\@sffamilyhook 430 \let\@sffamilyhook\@empty
\@ttfamilyhook 431 \let\@ttfamilyhook\@empty
\@defaultfamilyhook 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> \expand@font@defaults{Custom series with hooks}%
437 <latexrelease>
438 <latexrelease>\def\expand@font@defaults{%
439 <latexrelease> \edef\rmdef@ult{\rmdefault}%
440 <latexrelease> \edef\sfdef@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 <{\\latextitlepage}\\let\\emforce\\@undefined
583 <{\\latextitlepage}\\let\\em@force\\@undefined
584 <{\\latextitlepage}
585 <{\\latextitlepage}\\DeclareRobustCommand\\em
586 <{\\latextitlepage} {\\@nomath\\em \\ifdim \\fontdimen\\cne\\font >\\z@%
587 <{\\latextitlepage} \\eminnershape \\else \\itshape \\fi}%
588 <{\\latextitlepage}\\EndIncludeInRelease
589 <{\\latextitlepage}
590 <{\\latextitlepage}\\IncludeInRelease{0000/00/00}{\\DeclareEmphSequence}{Nested emph}%
591 <{\\latextitlepage}\\DeclareRobustCommand\\em
592 <{\\latextitlepage} {\\@nomath\\em \\ifdim \\fontdimen\\cne\\font >\\z@%
593 <{\\latextitlepage} \\upshape \\else \\itshape \\fi}%
594 <{\\latextitlepage}\\let\\eminnershape\\@undefined
595 <{\\latextitlepage}\\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 `newlfont.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^{`J}%
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 % {{\ooalign{\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\@addtofilelist##1{\xdef\@filelist{\@filelist,\###1}}%
750 }
751 {\input{fonttext.ltx}}
752 \let\@addtofilelist\@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\@addtofilelist##1{\xdef\@filelist{\@filelist,\###1}}%
760 }
761 {\input{fontmath.ltx}}
762 \let\@addtofilelist\@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\@addtofilelist##1{\xdef\@filelist{\@filelist,\###1}}%
770 }
771 {\input{preload.ltx}}
772 \let\@addtofilelist\@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.  
`\@accii`

```

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 A

## 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{2020/02/02}%
101 \let\updefault{\updefault{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
110 \let\IncludeInRelease{0000/00/00}%
111 \let\updefault{\updefault{font defaults change}}%
112 \let\renewcommand\updefault{n}
113 \let\renewcommand\bfdefault{bx}
114 \let\renewcommand\mddefault{undefined}
115 \let\renewcommand\EndIncludeInRelease
116 \let\bfdefault@previous\undefined
117 \let\mddefault@previous\undefined
118 \let\EndIncludeInRelease
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$}
560 \DeclareRobustCommand\upbracefill{$\m@th \setbox\z@\hbox{$\braceleft$}%
561 \braceleft\leaders\vrule \@height\ht\z@\@depth\z@\hfill\bracerd
562 \bracerd\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.

```

614 \DeclareMathSymbol{\mathparagraph}{\mathord}{symbols}{7B}
615 \DeclareMathSymbol{\mathsection}{\mathord}{symbols}{78}
616 \DeclareMathSymbol{\mathdollar}{\mathord}{operators}{24}
\mathsterling
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.\vphantom{1}\right. $}}\right.}\n@space$}}}
626 \DeclareRobustCommand{\Big[1]{\leavevmode@ifvmode
627 {\hbox{$\left.\vphantom{1}\right. $}}\right.}\n@space$}}}
628 \DeclareRobustCommand{\bigg[1]{\leavevmode@ifvmode
629 {\hbox{$\left.\vphantom{1}\right. $}}\right.}\n@space$}}}
630 \DeclareRobustCommand{\Bigg[1]{\leavevmode@ifvmode
631 {\hbox{$\left.\vphantom{1}\right. $}}\right.}\n@space$}}}
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 B

## 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.xpt | preload of CM fonts for 10pt document size                     |
| cmpreload.xip | preload of CM fonts for 11pt document size                     |
| cmpreload.xii | preload of CM fonts for 12pt document size                     |
| dcpreload.xpt | preload of DC fonts for 10pt size                              |
| dcpreload.xip | preload of DC fonts for 11pt size                              |
| dcpreload.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  $\text{\LaTeX}$  line and circle fonts. It is necessary to do this explicitly since these are used by the  $\text{\LaTeX}$  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 C

## 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}{\normalsize}`

(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@ic1 We define these defaults in case some error causes them to be expanded at the wrong time.  
\check@icr

```
44 \let \check@ic1 \@empty
45 \let \check@icr \@empty
```

(End definition for \check@ic1 and \check@icr.)

- \text@command This checks for a \nocorr as the first token in its argument and also for one in any other 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@ic1 \@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@ic1 \@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@ic1 \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@ic1 should be empty.

```
70 \let \check@ic1 \@empty
71 \else
```

Otherwise there is a \nocorr both at the start and elsewhere, so no italic corrections should be added.

```
72 \let \check@ic1 \@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@ictrue
```

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 \nocorlist
```

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 {,.}

```

(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{\langle font-change decls\rangle} {\langle math-alphabet\rangle}`

Here `\fn` is the font-declaration command being defined, `\langle font-change decls\rangle` is the declaration it will expand to in text-mode, and `\langle math-alphabet\rangle` 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 D

# lttextcomp.dtx

From File: 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` Preserve the old definition of `\oldstylenums` under a different name.

`\legacyoldstylenums` 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 0\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 \def\tc@swap@accent#1{\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{\textthreequartersdash}{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 list in the sections for sub-encodings  $x + 1$  and higher.

## 1.1 Sub-encoding 1 (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 Sub-encoding 2 (majority of new OTF fonts via autoinst)

```

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'}}
```

`\capitalbreve{u}`  
`\capitalcaron{v}`  
`\capitalcedilla{c}`  
`\capitalcircumflex{^}`  
`\capitaldieresis{"}`  
`\capitaldotaccent{.}`  
`\capitalgrave{@tabacckludge'}`  
`\capitalhungarumlaut{H}`  
`\capitalmacron{@tabacckludge=}`  
`\capitalnewtie{t}`  
`\capitalogonek{k}`  
`\capitalring{r}`  
`\capitaltie{t}`  
`\capitaltilde{\~}`  
`\newtie{t}`  
199 \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 -OsF, 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{\texteightholdstyle}
215 {\tc@check@symbol2\texteightholdstyle}
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{\textasciacute}
237 {\tc@check@symbol2\textasciacute}
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 {\tc@check@symbol2\textbigcircle}
256 \DeclareTextCommandDefault{\textborn}
257 {\tc@check@symbol2\textborn}

```

```

258 \DeclareTextCommandDefault{\textcentoldstyle}
259 {\tccheck@symbol2\textcentoldstyle}
260 \DeclareTextCommandDefault{\textcircledP}
261 {\tccheck@symbol2\textcircledP}
262 \DeclareTextCommandDefault{\textcopyleft}
263 {\tccheck@symbol2\textcopyleft}
264 \DeclareTextCommandDefault{\textdblhyphenchar}
265 {\tccheck@symbol2\textdblhyphenchar}
266 \DeclareTextCommandDefault{\textdblhyphen}
267 {\tccheck@symbol2\textdblhyphen}
268 \DeclareTextCommandDefault{\textdied}
269 {\tccheck@symbol2\textdied}
270 \DeclareTextCommandDefault{\textdiscount}
271 {\tccheck@symbol2\textdiscount}
272 \DeclareTextCommandDefault{\textdivorced}
273 {\tccheck@symbol2\textdivorced}
274 \DeclareTextCommandDefault{\textdollaroldstyle}
275 {\tccheck@symbol2\textdollaroldstyle}
276 \DeclareTextCommandDefault{\textguarani}
277 {\tccheck@symbol2\textguarani}
278 \DeclareTextCommandDefault{\textleaf}
279 {\tccheck@symbol2\textleaf}
280 \DeclareTextCommandDefault{\textlquill}
281 {\tccheck@symbol2\textlquill}
282 \DeclareTextCommandDefault{\textmarried}
283 {\tccheck@symbol2\textmarried}
284 \DeclareTextCommandDefault{\textmho}
285 {\tccheck@symbol2\textmho}
286 \DeclareTextCommandDefault{\textmusicalnote}
287 {\tccheck@symbol2\textmusicalnote}
288 \DeclareTextCommandDefault{\textnaira}
289 {\tccheck@symbol2\textnaira}
290 \DeclareTextCommandDefault{\textopenbullet}
291 {\tccheck@symbol2\textopenbullet}
292 \DeclareTextCommandDefault{\textpeso}
293 {\tccheck@symbol2\textpeso}
294 \DeclareTextCommandDefault{\textpilcrow}
295 {\tccheck@symbol2\textpilcrow}
296 \DeclareTextCommandDefault{\textrecipe}
297 {\tccheck@symbol2\textrecipe}
298 \DeclareTextCommandDefault{\textremark}
299 {\tccheck@symbol2\textremark}
300 \DeclareTextCommandDefault{\textrquill}
301 {\tccheck@symbol2\textrquill}
302 \DeclareTextCommandDefault{\textservicemark}
303 {\tccheck@symbol2\textservicemark}
304 \DeclareTextCommandDefault{\textsurd}
305 {\tccheck@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 Sub-encoding 3

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 Sub-encoding 4

```
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 Sub-encoding 5 (most older PS fonts)

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 Sub-encoding 6

```
333 \DeclareTextCommandDefault{\textflorin}
334 {\tc@check@symbol6\textflorin}
335 \DeclareTextCommandDefault{\textcurrency}
336 {\tc@check@symbol6\textcurrency}
```

## 1.7 Sub-encoding 7

```
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 Sub-encoding 8

```
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 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> {ltxtextcomp}{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`\'$\\egroup}
632 <|latexrelease>\DeclareTextCommand{\textsterling}{\OT1}{\hmode@bgroup
633 <|latexrelease> \ifdim \fontdimen1ne\font >\z@
634 <|latexrelease> \itshape
635 <|latexrelease> \else
636 <|latexrelease> \fontshape{ui}\selectfont
637 <|latexrelease> \fi
638 <|latexrelease> \char`\'$\\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-TEX 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  $\backslash\langle arg2\rangle:\langle current-family\rangle$  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  $\backslash\CheckEncodingSubset$ .)

$\backslash\textc@subst$

```

911 \def\textc@subst#1{%
912 \textc@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\textcompsubstddefault\selectfont#1\egroup
917 }

```

(End definition for  $\backslash\textc@subst$ .)

$\backslash\textc@error$

$\backslash\textc@error$  is going to be used in arg #3 of  $\backslash\CheckEncodingSubset$  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\textc@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  $\backslash\textc@error$ .)

$\backslash\textc@fake@euro$

$\backslash\textc@fake@euro$  is an example of a “fake” definition to use in arg #3 of  $\backslash\CheckEncodingSubset$  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\textc@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 E

## 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 F

## 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}{{'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 `\else`. 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 G

## 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.)*

**\enddocument**

```

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
```

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.

```
61 \fi
62 }
```

*(End definition for `\@enddocument@kernel@warnings`.)*

```
63 </2ekernel | latexrelease>
64 <latexrelease>\EndIncludeInRelease
65 <latexrelease>\IncludeInRelease{0000/00/00}%
66 <latexrelease> {\@enddocument}{Use Hooks}%
67 <latexrelease>
68 <latexrelease>\def\enddocument{%
69 <latexrelease> \let\AtEndDocument\@firstofone
70 <latexrelease> \Cenddocumenthook
71 <latexrelease> \Ccheckend{document}%
72 <latexrelease> \clearpage
73 <latexrelease> \begingroup
74 <latexrelease> \if@filesw
75 <latexrelease> \immediate\closeout\mainaux
76 <latexrelease> \let\@setckpt\@gobbletwo
77 <latexrelease> \let\@newlbel\@testdef
78 <latexrelease> \tempswafalse
79 <latexrelease> \makeatletter \C@input\jobname.aux
80 <latexrelease> \fi
81 <latexrelease> \Cdo filelist
82 <latexrelease> \ifdim \font@submax >\fontsubfuzz\relax
83 <latexrelease> \Cfont@warning{Size substitutions with differences\MessageBreak
84 <latexrelease> up to \font@submax\space have occurred.\@gobbletwo}%
```

```

85 〈latexrelease〉 \fi
86 〈latexrelease〉 \@defaultsubs
87 〈latexrelease〉 \@refundefined
88 〈latexrelease〉 \if@filesw
89 〈latexrelease〉 \ifx \c@multiplelabels \relax
90 〈latexrelease〉 \if@tempswa
91 \c@latex@warning@no@line{Label(s) may have changed.
92 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〉 {\c@kernel@before@cenddocument}{kernel before hook}%
109 \def\c@kernel@before@cenddocument{\par}
110 〈/2ekernel | latexrelease〉
111 〈latexrelease〉\EndIncludeInRelease

```

The rollback code renders it harmless.

```

112 〈latexrelease〉\IncludeInRelease{0000/00/00}%
113 〈latexrelease〉 {\c@kernel@before@cenddocument}{kernel before hook}%
114 〈latexrelease〉
115 〈latexrelease〉\let\c@kernel@before@cenddocument\c@empty
116 〈latexrelease〉
117 〈latexrelease〉\EndIncludeInRelease
118 〈*2ekernel〉

```

(End definition for \@kernel@before@cenddocument.)

```

\@testdef
119 \def\c@testdef #1#2#3{%
120 \def\c@reserved@a{#3}\expandafter \ifx \csname #1#2\endcsname
121 \c@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 \@ifundefined{tf@#1}\relax
160 {\@temptokena{#2}%
161 \immediate\write\csname tf@#1\endcsname{\the\@temptokena}%
162 }%
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 \ignorespaces
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 \@namedef{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}

```

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 195.*)

## 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** 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 We use \relax to prevent \item scanning too far.
401 \def\flushleft{\trivlist \raggedright\item\relax}
402 \def\endflushleft{\endtrivlist}

flushright 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 T<sub>E</sub>Xbook's space character instead of as blank spaces.

\@vobeyspaces

406 {\catcode`\\ =\active%

407 \gdef\@vobeyspaces{\catcode`\\ \active\let \obeysp{}}

(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{\end{verbatim}[\#1]\end{verbatim}}

411 \gdef\@sxverbatim{\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> \DisableHyphenationInVerbatim %

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`.)*

**`\verbatim`** (RmS 93/09/19) Protected against ‘missing item’ error message triggered by empty verbatim environment.

```

459 \def\verbatim{\overbatim \frenchspacing\obeyspaces \xoverbatim}
460 \def\endverbatim{\if@newlist \leavevmode\fi\endtrivlist}

```

*(End definition for `\verbatim` and `\endverbatim`.)*

**`\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 \verbatim@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\*** 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\o@empty
```

(End definition for \verb@balance@group.)

\verb@egroup

```
547 \def\verb@egroup{\global\let\verb@balance@group\o@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 H

## 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
```

`\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.

```

201 〈/2ekernel〉
202 〈*2ekernel | latexrelease〉
203 〈latexrelease〉\IncludeInRelease{2020/10/01}%
204 〈latexrelease〉 {\tmspace}{amsmath spacing commands}%
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 Disguises for \(\dots\) and \[\dots\].
displaymath
332 \let\math=\(
333 \let\endmath=\)
334 \def\displaymath{[]}
335 \def\enddisplaymath[]\@ignoretrue

\c@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 \c@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 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
\c@eqnsw
344 \newcount\c@eqcnt
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\@eqnse=\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 \@@eqncr
391 \noalign{\penalty\eqpen\vskip\jot\@vspace\calcify{#1}}%
392 }
393 </2ekernel | latexrelease>
394 <latexrelease>\EndIncludeInRelease
395 <latexrelease>\IncludeInRelease{0000/00/00}%
396 <latexrelease> {\@xeqncr}{eqnarray support calc syntax}%
397 <latexrelease>
398 <latexrelease>\def\@xeqncr[#1]{%
399 <latexrelease> \ifnum0={\fi}%
400 <latexrelease> \@@eqncr
401 <latexrelease> \noalign{\penalty\eqpen\vskip\jot\vskip #1\relax}%
402 <latexrelease>}
403 <latexrelease>\EndIncludeInRelease
404 <2ekernel>

```

(End definition for `\@eqncr`, `\@xeqncr`, and `\@yeqncr`.)

```

\@@eqncr
405 \def\@@eqncr{\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 `\@@eqncr`.)

**eqnarray\*** Here's the eqnarray\* environment:

```

412 \let\@seqncr=\@eqncr
413 \@namedef{eqnarray*}{\protected\def\@eqncr{\nonumber\@seqncr}\eqnarray}
414 \@namedef{endeqnarray*}{\nonumber\endeqnarray}

```

(End definition for `\@seqncr`.)

**\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\normalcolor}`  
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 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** 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 I

## 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

\begin{parpenalty}: 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 @noperitem = true
then @noperitem := false
 % NOTE: then clause hardly every taken,
 % so made a macro \odonoperitem
\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
\leftmarginiii 18 \newdimen\leftmarginii
\leftmarginiv 19 \newdimen\leftmarginiii
\leftmarginv 20 \newdimen\leftmarginiv
\leftmarginvi 21 \newdimen\leftmarginv
22 \newdimen\leftmarginvi

(End definition for \leftmargini and others.)

\@listdepth
\@itempenalty 23 \newcount\@listdepth \@listdepth=0
\@beginparpenalty 24 \newcount\@itempenalty
\@endparpenalty 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@noparlist
\@noparlistfalse 31 \newif\if@noparlist \@noparlistfalse
\@noparlisttrue
(End definition for \if@noparlist, \@noparlistfalse, and \@noparlisttrue.)

\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
 34 \def\list#1#2{%
 35 \ifnum \clistdepth >5\relax
 36 \toodeep
 37 \else
 38 \global\advance\clistdepth\one
 39 \fi
 40 \rightmargin\z@%
 41 \listparindent\z@%
 42 \itemindent\z@%
 43 \csname @list\romannumeral\the\clistdepth\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 \@totalleftmargin \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 \setpart{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.)

```

\textrivlist
89 \def\textrivlist{%
90 \parsep\parskip
91 \cnewlistfalse
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 ltspace.dtx so that other commands could be ‘let to it’. They now use \def.

```

\endtrivlist
101 \def\endtrivlist{%
102 \if@inlabel
103 \leavevmode
104 \global \cnewlistfalse
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 LATEX 2.09 comments.

\c@enumdepth
226 \newcount\c@enumdepth \c@enumdepth = 0

(End definition for \c@enumdepth.)

\c@enumi
\c@enumii 227 \c@definecounter{enumii}
\c@enumiii 228 \c@definecounter{enumiii}
\c@enumiv 229 \c@definecounter{enumiv}
230 \c@definecounter{enumiv}

(End definition for \c@enumi and others.)

enumerate
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 LATEX 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 LATEX 2.09 comments.
```

```

\@itemdepth
241 \newcount\@itemdepth \@itemdepth = 0
(End definition for \@itemdepth.)
```

```

itemize
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 J

## 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 \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` added 20 Jan 87  
`\ 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` : 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`  
`\underline`  
`\raisebox`  
`\rule[<raised>]{<width>}{<height>} : Makes a <width>*<height> rule, raised <raised>.`  
`\underline{<text>} : Makes an underlined hbox with <text> in it.`  
`\raisebox{<distance>}{<height>}[<depth>]{<box>} :`  
 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` User level command just looks for optional [ or .  
`3 (/2ekernel)`  
`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 \atfor\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 \vbox to#2\unitlength
74 {\let\mb@b\vss \let\mb@l\hss\let\mb@r\hss
75 \let\mb@t\vss
76 \atfor\reserved@a :=#3\do{%
77 \if s\reserved@a
78 \let\mb@l\relax\let\mb@r\relax
79 \else
80 \expandafter\let\csname mb@\reserved@a\endcsname\relax
81 \fi}%
82 \mb@t
83 \hb@xt@ #1\unitlength{\mb@l #4\mb@r}%
84 \mb@b
85 \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 \@endpefalse
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 {\@ifframebox[#1]}%
200 {\@ifframebox[#1][c]}}

```

(End definition for \@framebox.)

\@ifframebox 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\@ifframebox[#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 \@ifframebox.)

\@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]{%
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{#1\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 |> \def\@arrayparboxrestore{%
285 \let\ifnobreak\iffalse
286 \let\ifnoskipsec\iffalse
287 \let\par\@par
288 \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\mpelistdepth \c@mpelistdepth\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 `\@listdepth` 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 \global\setbox\@mpfootins\vbox{%
393 \unvbox\@mpfootins
394 \reset@font\footnotesize
395 \hsize\columnwidth
396 \parboxrestore
397 \protected@edef\@currentlabel
398 {\csname p@\mpfootnote\endcsname\@thefnmark}%
399 \color@begingroup
400 \makefntext{%
401 \rule{z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
402 \par
403 \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 \global\setbox\@mpfootins\vbox{%
410 \unvbox\@mpfootins
411 \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`  
`\centerline`  
`\@oline`  
`\@coline`  
`\@clline`

```
489 \def\@cline{\hb@xt@\hsize}
490 \DeclareRobustCommand\leftline[1]{\@cline{#1\hss}}
491 \DeclareRobustCommand\rightline[1]{\@cline{\hss#1}}
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 K

## 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\@tabfbox = \rlap{\indent\the\everypar}
 % note: \the\everypar sets @inlabel :=G F
 \@itemfudge == BEGIN \box\@tabfbox 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 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 \@latex@error{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 \@latex@error{\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

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\*.*)

\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{\dimen0{#1}}%  
177 \edef\@haligno{to\the\dimen0}\@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\@ END

\@xexpast S1 *{N}{S2} S3 \@@ ==
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 \@@
END
End of historical LATEX 2.09 comments.

\@xexnoop
238 \def\@xexnoop #1\@{}{}

(End definition for \@xexnoop.)

\@expast
239 \def\@expast#1{\@xexpast #1*0x\@{}}

(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`.)

```
\@carstrutbox
 \@carstrut 343 \newbox\@carstrutbox
 344 \def\@carstrut{
 345 \relax\ifmmode\copy\@carstrutbox\else\unhcopy\@carstrutbox\fi}
```

(End definition for `\@carstrutbox` and `\@carstrut`.)

```
\@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
357 }
```

(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 L

## 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 [⟨N⟩] ⟨⟨AX,AY⟩⟩⟨⟨BX,BY⟩⟩⟨⟨CX,CY⟩⟩ plots a quadratic Bezier curve from ⟨⟨AX,AY⟩⟩ to ⟨⟨CX,CY⟩⟩, with ⟨⟨BX,BY⟩⟩ 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.

```
 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>
```

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.

```
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.)

**\picture** #1 should be white space.

#1 should be a ( (eating any white space before the bracket),

```
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\bgroup
32 \@defaultunitsset\@tempdimc{#3}\unitlength
33 \hskip -\@tempdimc
34 \@defaultunitsset\@tempdimc{#4}\unitlength
35 \lower\@tempdimc\hbox\bgroup
36 \ignorespaces}
37 </2ekernel | latexrelease>
```

```

38 \end{macro}
39 \end{macro}
40 \def\@picture#1#2#3#4{\%
41 \begingroup\hbox{#1}\unitlength\begin{array}{c}#2\\#3\\#4\end{array}\endgroup
42 \raisebox{#3\unitlength}{#2}
43 \ignorespaces}
44 \end{macro}
45 \end{macro}
46 \end{macro}
47 \end{macro}
48 \end{macro}

```

(End definition for `\@picture`.)

```

\end{macro}

```

```

49 \def\endpicture{\%
50 \egroup\hss\egroup
51 \ht\@picbox\@picht\dp\@picbox\z@%
52 \mbox{\box\@picbox}}

```

(End definition for `\endpicture`.)

In the definitions of `\put` and `\multiput`, `\hskip` 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 \def\multiput{#1#2}{%
82 \expandafter\let\csname multiput \endcsname\undefined
83 \def\multiput(#1,#2){%
84 \xdef\multiput{\xdef\multiput{\#1}\unitlength
85 \xdef\multiput{\#2}\unitlength
86 \multiput{}}
87 }/2ekernel | latexrelease}
88 \EndIncludeInRelease
89 \IncludeInRelease{0000/00/00}%
90 \def\multiput{#1#2}{%
91 \expandafter\let\csname multiput \endcsname\undefined
92 \def\multiput(#1,#2){%
93 \xdef\multiput{\#1}\unitlength
94 \xdef\multiput{\#2}\unitlength
95 \multiput{}}
96 \EndIncludeInRelease
97 }*2ekernel}

```

(*End definition for \multiput.*)

```

\@multiput
98 /2ekernel}
99 {*2ekernel | latexrelease}
100 \IncludeInRelease{2020/10/01}%
101 \def\multiput{#1#2}{%
102 \long\def\multiput(#1,#2){%
103 \killglue\multicnt #3\relax
104 \whilenum \multicnt >\z@\do
105 {\raise\ydim\hb@xt@.z@{\kern\xdim #4\hss}%
106 \advance\multicnt\m@ne
107 \defaultunits{\advance\xdim}{#1}\unitlength
108 \defaultunits{\advance\ydim}{#2}\unitlength}%
109 \ignorespaces}
110 }/2ekernel | latexrelease}
111 \EndIncludeInRelease
112 \IncludeInRelease{0000/00/00}%
113 \def\multiput{#1#2}{%
114 \long\def\multiput(#1,#2){%
115 \killglue\multicnt #3\relax
116 \whilenum \multicnt >\z@\do
117 {\raise\ydim\hb@xt@.z@{\kern\xdim #4\hss}%
118 \advance\multicnt\m@ne
119 \advance\xdim#1\unitlength\advance\ydim#2\unitlength}%
120 \ignorespaces}
121 \EndIncludeInRelease
122 }*2ekernel}

```

(*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\bgrou
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@\@badlinearg\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@\@badlinearg\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 \yyarg >\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@{z0}{\@linefnt
266 \ifnum \oxarg < z0 \@getlarrow(1,0)\hss\else
267 \hss\@getrarrow(1,0)\fi}}
(End definition for \@hvector.)
```

```

\@vvector
268 \def\@vvector{\ifnum \oyarg < z0 \@downvector \else \@upvector \fi}
(End definition for \@vvector.)
```

```

\@svector
269 \def\@svector{\@sline
270 \tempcnta\@yarg \ifnum\tempcnta < z0 \tempcnta -\tempcnta\fi
271 \ifnum\tempcnta < 5%
272 \hskip -\wd\@linechar
273 \upordown\clnht \hbox{\@linefnt \if@negarg
274 \@getlarrow(\oxarg,\yyarg)\else \@getrarrow(\oxarg,\yyarg)\fi}%
275 \else\badlinearg\fi}
(End definition for \@svector.)
```

```

\@getlarrow
276 \def\@getlarrow(#1,#2){\ifnum #2=z0 \tempcnta 27 % '33
277 \else
278 \tempcnta #1\relax\multiply\tempcnta \sixt@n
279 \advance\tempcnta -9 \tempcntb #2\relax\multiply\tempcntb \tw@0
280 \ifnum \tempcntb > z0 \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 < z0 \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< z0 \advance\tempcnta 64 \fi
296 \char\tempcnta}
(End definition for \@getrarrow.)
```

```

\@vline
297 \def\@vline{\ifnum \oyarg < z0 \@downline \else \@upline\fi}
```

(End definition for \@vline.)

```
\@upline
298 \def\@upline{%
299 \hb@xt@z@{\hskip -\@halfwidth \vrule \@width \@wholewidth
300 \@height \@linelen \@depth \z@\hss}}
301
```

(End definition for \@upline.)

\@downline

```
301 \def\@downline{%
302 \hb@xt@z@{\hskip -\@halfwidth \vrule \@width \@wholewidth
303 \@height \z@ \@depth \@linelen \hss}}
304
```

(End definition for \@downline.)

\@upvector

```
304 \def\@upvector{\@upline\setbox\@tempboxa\hbox{\@linefnt\char 54}\% '66
305 \raise \@linelen \hb@xt@z@\lower \ht\@tempboxa\box\@tempboxa\hss}
306
```

(End definition for \@upvector.)

\@downvector

```
306 \def\@downvector{\@downline\lower \@linelen
307 \hb@xt@z@\@linefnt\char 63 \% '77
308 \hss]}
309
```

(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 L<sup>A</sup>T<sub>E</sub>X 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 \tempcpta\z@
361 \put(0,0){\hskip -\@halfwidth \vbox{\@whilenum \tempcpta <\@dashcnt
362 \do{\vskip\@tempdimb\copy\@dashbox\advance\tempcpta \one }%
363 \vskip\@dashdim}}\tempcpta\z@
364 \put(#2,0){\hskip -\@halfwidth \vbox{\@whilenum \tempcpta<\@dashcnt
365 \do{\vskip\@tempdimb\copy\@dashbox\advance\tempcpta \one }%
366 \vskip\@dashdim}}\makepicbox(#2,#3)}
367 </ekernel | latexrelease>

368 <texrelease>\EndIncludeInRelease
369 <texrelease>\IncludeInRelease{0000/00/00}%
370 <texrelease> {\dashbox}{default units}%
371 <texrelease>\expandafter\let\csname dashbox \endcsname\undefined
372 <texrelease>\def\dashbox#1{#2,#3}{%
373 <texrelease>\leavevmode\hb@xt@z@{\baselineskip \z@skip
374 <texrelease>\lineskip \z@skip
375 <texrelease>\@dashdim #2\unitlength
376 <texrelease>\@dashcnt \@dashdim \advance\@dashcnt 200
377 <texrelease>\@dashdim #1\unitlength\divide\@dashcnt \@dashdim
378 <texrelease>\ifodd\@dashcnt\@dashdim \z@
379 <texrelease>\advance\@dashcnt \one \divide\@dashcnt \tw@
380 <texrelease>\else \divide\@dashdim \tw@ \divide\@dashcnt \tw@
381 <texrelease>\advance\@dashcnt \m@ne
382 <texrelease>\setbox\@dashbox\hbox{%
383 <texrelease> \vrule \height \@halfwidth \depth \@halfwidth
384 <texrelease> \width \@dashdim}\put(0,0){\copy\@dashbox}%
385 <texrelease>\put(0,#3){\copy\@dashbox}%
386 <texrelease>\put(#2,0){\hskip-\@dashdim\copy\@dashbox}%
387 <texrelease>\put(#2,#3){\hskip-\@dashdim\box\@dashbox}%
388 <texrelease>\multiply\@dashdim \thr@@
389 <texrelease>\fi
390 <texrelease>\setbox\@dashbox\hbox{%
391 <texrelease> \vrule \height \@halfwidth \depth \@halfwidth
392 <texrelease> \width #1\unitlength\hskip #1\unitlength}\tempcpta\z@
393 <texrelease>\put(0,0){\hskip\@dashdim \@whilenum \tempcpta <\@dashcnt
394 \do{\copy\@dashbox\advance\tempcpta \one }}\tempcpta\z@
395 <texrelease>\put(0,#3){\hskip\@dashdim \@whilenum \tempcpta <\@dashcnt
396 \do{\copy\@dashbox\advance\tempcpta \one }}%
397 <texrelease>\@dashdim #3\unitlength
398 <texrelease>\@dashcnt \@dashdim \advance\@dashcnt 200
399 <texrelease>\@dashdim #1\unitlength\divide\@dashcnt \@dashdim
400 <texrelease>\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 \tfor\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 -2pt
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} {\@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 `\@ovhorz`.)

```
\circle
605 \def\circle{\c@inmatherr\circle\@ifstar\@dot\@circle}
```

(End definition for `\circle`.)

```
\@circle
606 {/2ekernel}
607 {*2ekernel | latexrelease}
608 \end{latexrelease} \IncludeInRelease{2020/10/01}%
609 \end{latexrelease} {\@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{\@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} {\@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 \@xdim := |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@\z@}{\@tempcnta}\do
788 {\@xdim\count@\@ovxx
789 \advance\@xdim\@ovdx

```

```

790 〈\latexrelease〉 \divide\@xdim\@multicnt
791 〈\latexrelease〉 \multiply\@xdim\count@
792 〈\latexrelease〉 \ydim\count@\@ovyy
793 〈\latexrelease〉 \advance\ydim\@ovdy
794 〈\latexrelease〉 \divide\ydim\@multicnt
795 〈\latexrelease〉 \multiply\ydim\count@
796 〈\latexrelease〉 \raise\ydim
797 〈\latexrelease〉 \hb@xt@z@{\kern\@xdim
798 〈\latexrelease〉 \unhcopy\tempboxa\hss}%
799 〈\latexrelease〉 \advance\count@\@ne}}}
800 〈\latexrelease〉\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 〈/2ekernel〉
803 〈*2ekernel | \latexrelease〉
804 〈\latexrelease〉\IncludeInRelease{2019/10/01}%
805 〈\latexrelease〉 {\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 〈/2ekernel | \latexrelease〉
819 〈\latexrelease〉\EndIncludeInRelease
820 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
821 〈\latexrelease〉 {\bezier}{\Make commands robust}%
822 〈\latexrelease〉
823 〈\latexrelease〉\kernel@make@fragile\bezier
824 〈\latexrelease〉\kernel@make@fragile\circle
825 〈\latexrelease〉\kernel@make@fragile\dashbox
826 〈\latexrelease〉\kernel@make@fragile\line
827 〈\latexrelease〉\kernel@make@fragile\linethickness
828 〈\latexrelease〉\kernel@make@fragile\multiput
829 〈\latexrelease〉\kernel@make@fragile\oval
830 〈\latexrelease〉\kernel@make@fragile\put
831 〈\latexrelease〉\kernel@make@fragile\qbezier
832 〈\latexrelease〉\kernel@make@fragile\shortstack
833 〈\latexrelease〉\kernel@make@fragile\thinlines
834 〈\latexrelease〉\kernel@make@fragile\vector
835 〈\latexrelease〉
836 〈\latexrelease〉\EndIncludeInRelease
837 〈*2ekernel〉
838 〈/2ekernel〉

```

# File M

## 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

`\@thmcnter{COUNTER}` : A command such that

`\edef\theCOUNTER{\@thmcnter{COUNTER}}`

defines `\theCOUNTER` to produce a number for a theorem environment.

The default is:

`BEGIN \noexpand\arabic{COUNTER} END`

`\@thmcntersep` : 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, `\@thmcntersep` should be `\def`’ed to ‘-’. 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 N

## 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>}`. 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 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{\langle name \rangle}{\langle level \rangle}{\langle indent \rangle}{\langle beforeskip \rangle}{\langle afterskip \rangle}{\langle style \rangle}*` 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

**beforeskip:** 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 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\csname tf@#1\endcsname
155 \immediate\openout \csname 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 {latexrelease}\IncludeInRelease{2018/12/01}%
203 {latexrelease} {\@dottedtocline}{Prevent protrusion}%
204 \def\@dottedtocline#1#2#3#4#5{%
205 \ifnum #1>\c@tocdepth \else
206 \vskip \z@ \oplus .2\p@
207 {\leftskip #2\relax \rightskip \c@tocrmarg \parfillskip -\rightskip
208 \parindent #2\relax\@afterindenttrue
209 \interlinepenalty\@M
210 \leavevmode
211 \c@tempdima #3\relax
212 \advance\leftskip \c@tempdima \null\nobreak\hskip -\leftskip
213 {#4}\nobreak
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 \hb@xt@\c@pnumwidth{\hfil\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 O

## 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.

\fptype@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*\fptype@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
3 \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 \@nodocument
40 \def \@capttype {#1}%
41 \def \@fps {#2}%
42 \onelevel@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 \ifinner
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 \fltofvf
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> \noldocument
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@
128 <|latexrelease> \else
129 <|latexrelease> \cnext\currbox\freelist
130 <|latexrelease> {%
131 <|latexrelease> \tempcnta \sixt@@n
132 <|latexrelease> \expandafter \tfor \expandafter \reserved@a
133 <|latexrelease> \expandafter :\expandafter =\fps
134 <|latexrelease> \do
135 <|latexrelease> {%
136 <|latexrelease> \if \reserved@a h%
137 <|latexrelease> \ifodd \tempcnta
138 <|latexrelease> \else
139 <|latexrelease> \advance \tempcnta \one
140 <|latexrelease> \fi
141 <|latexrelease> \fi
142 <|latexrelease> \if \reserved@a t%
143 <|latexrelease> \setfpsbit \tw@
144 <|latexrelease> \fi
145 <|latexrelease> \if \reserved@a b%
146 <|latexrelease> \setfpsbit 4%
147 <|latexrelease> \fi
148 <|latexrelease> \if \reserved@a p%
149 <|latexrelease> \setfpsbit 8%
150 <|latexrelease> \fi
151 <|latexrelease> \if \reserved@a !%
152 <|latexrelease> \ifnum \tempcnta>15
153 <|latexrelease> \advance\tempcnta -\sixt@@n\relax
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> \fltofvf
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\@dbldeflist\@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\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:  
This should be called whenever \@colht has been set.

```

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
```

(End definition for \@floatplacement.)

\@dblfloatplacement This should be called only within a group. Now changed to provide extra checks in \addtodblcol, needed when processing a BANG float.

```

280 <texreleas>\IncludeInRelease{2015/01/01}%
281 <texreleas> {\@dblfloatplacement}{float order in 2-column}%
282 {*2ekernel | texreleas>}

```

When making two column float area, look for floats with 1sp depth.

```

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

```

\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.

```

291 \def\f@depth{1sp}%
292
```

Textpage bit: global, but need not be.

```

297 <texreleas> \global\@dbltopnum\c@dbltopnumber
298 <texreleas> \global\@dbltoproom \dbltopfraction\@colht

```

This new bit uses \@textmin to locally store the amount of extra room in the column.

```

299 <texreleas> \@textmin \@colht
300 <texreleas> \advance \@textmin -\@dbltoproom

```

Floatpage bit: must be local.

```

301 <texreleas> \@fpmin \dblfloatpagefraction\textheight
302 <texreleas> \@fptop \@dblfpptop
303 <texreleas> \@fpsep \@dblfpsep
304 <texreleas> \@fpbot \@dblfpbot
305 <texreleas>}%
306 <texreleas>\EndIncludeInRelease
307 {*2ekernel}

```

(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@ltovf\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 <latexrelease> \global\setbox #1%
 357 <latexrelease> \color@vbox
 358 <latexrelease> \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 \ignorespaces
385 \esphack
386 }
```

(End definition for `\@xympar`.)

### \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
\@makefnmark % put number in main text
IF hmode THEN \spacefactor := \c@sf FI
END

\footnotetext ==
BEGIN begingroup \protect == \noexpand
\@thefnmark :=G eval (\thempfn)
endgroup
\@footnotetext
END

\footnotetext[NUM] ==
BEGIN begingroup counter \c@mpfn :=L NUM
\protect == \noexpand
\@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 \Qdefinecounter{footnote}

397 \def\thefootnote{\@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 \Qdefinecounter{mpfootnote}

399 \def\thempfootnote{\itshape\@alph\c@mpfootnote}}

(*End definition for \thempfootnote.*)

\@makefnmark Default definition.

```

400 \%def\@makefnmark{\hbox{$^{\@thefnmark}\m@th$}}
401 \def\@makefnmark{\hbox{\@textsuperscript{\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 </2ekernel>
405 <*2ekernel | latexrelease>
406 <latexrelease>\IncludeInRelease{2020/10/01}%
407 <latexrelease> {\@textsuperscript}{superscript baseline}%
408 \def\@textsuperscript#1{%
409 {\m@th\ensuremath{^{\mbox{\scriptsize\sffamily#1}}}}}
410 </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{\scriptsize\sffamily#1}}}}}
417 <latexrelease>\EndIncludeInRelease
418 <*2ekernel>

```

(End definition for \@textsuperscript.)

\textsubscript

```

419 </2ekernel>
420 <latexrelease>\IncludeInRelease{2015/01/01}%
421 <latexrelease> {\textsubscript}{\textsubscript}%
422 <*2ekernel | latexrelease>
423 \DeclareRobustCommand*\textsubscript[1]{%
424 \@textsubscript{\selectfont#1}}%
425 </2ekernel | latexrelease>
426 <latexrelease>\EndIncludeInRelease
427 <latexrelease>\IncludeInRelease{0000/00/00}%
428 <latexrelease> {\textsubscript}{\textsubscript}%
429 <latexrelease>\let\textsubscript@\undefined
430 <latexrelease>\EndIncludeInRelease
431 <*2ekernel>

```

(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 <latexrelease> {\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}}
```

(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}
```

(End definition for \@xfootnote.)

\@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 P

## 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.
```

<sup>1</sup> {\*2ekernel}  
<sup>2</sup> \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 Q

## 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.

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
 3 \def\bibitem{\@ifnextchar[\@lbibitem\@bibitem}
 (End definition for \bibitem.)

\@lbibitem
 4 \def@\lbibitem[#1]#2{\item[\@biblabel{#1}\hfill]\if@filesw
 5 {\let\protect\noexpand
 6 \immediate
 7 \write\auxout{\string\bibcite{#2}{#1}}}\fi\ignorespaces}
 (End definition for \@lbibitem.)

\@bibitem
 8 \def@\bibitem#1{\item\if@filesw \immediate\write\auxout
 9 {\string\bibcite{#1}{\the\value{@listctr}}}\fi\ignorespaces}
 (End definition for \@bibitem.)

\bibcite
10 \def\bibcite{\newl@bel b}
 (End definition for \bibcite.)

\citation
11 \let\citation@gobble
 (End definition for \citation.)

\cite
12 \DeclareRobustCommand\cite{%
13 \@ifnextchar [{\@tempswatrue\@citex}{\@tempswafalse\@citex[]}}
 (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)
14 \def@\citex[#1]#2{\leavevmode
15 \let@\citea\@empty
16 \@cite{\@for\@cited:=#2\do
17 {\@citea\def@\citea{,\penalty\@m\ }}%
18 \edef@\citeb{\expandafter\@firstofone\@cited\@empty}%
19 \if@filesw\immediate\write\auxout{\string\citation{\@cited}}\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.

```

20 \ifundefined{b@\@citeb}{\hbox{\reset@font\bfseries ?}}%
21 \G@refundefinedtrue
22 \G@latex@warning
23 {Citation ‘@\@citeb’ on page \thepage \space undefined}%
24 {\@cite@ofmt{\csname b@\@citeb\endcsname}}}\#1}

```

(End definition for `\@citex`.)

```

\bibdata
\bibstyle 25 \let\bibdata=\@gobble
26 \let\bibstyle=\@gobble

```

(End definition for `\bibdata` and `\bibstyle`.)

```

\bibliography
27 \def\bibliography#1{%
28 \if@filesw
29 \immediate\write\auxout{\string\bibdata{\zap@space#1 \empty}%
30 \fi
31 \@input{\jobname.bbl}}

```

(End definition for `\bibliography`.)

```

\bibliographystyle
32 \def\bibliographystyle#1{%
33 \ifx\@begindocumenthook\undefined\else
34 \expandafter\AtBeginDocument
35 \fi
36 {\if@filesw
37 \immediate\write\auxout{\string\bibstyle{#1}}%
38 \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 `\@citex`, to get rid of leading spaces.

```

39 </2ekernel>
40 <*2ekernel | latexrelease>
41 <latexrelease>\IncludeInRelease{2021/06/01}%
42 <latexrelease> {\nocite}{Allow nocite in preamble}%
43 \def\nocite#1{\@bsphack

```

With the implementation designed already in L<sup>A</sup>T<sub>E</sub>X 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 L<sup>A</sup>T<sub>E</sub>X, missing all citations done with `\nocite`. Thus we do only generate an error message and leave the fix for a L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub>  successor.]

Given that we are now a quarter century into using L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub>  there is no good reason any more do limit ourself to 2.09 conciderations. So we now simply delay the `\nocite` if it is issued in the preamble.

```
44 \ifx\@onlypreamble\document
```

Since we are after `\begin{document}` we can do the citations:

```
45 \@for\@citeb:=#1\do{%
46 \edef\@citeb{\expandafter\@firstofone\@citeb}%
47 \if@filesw\immediate\write\@auxout{\string\citation{\@citeb}}\fi
48 \@ifundefined{b@\@citeb}{\G@refundefinedtrue
49 \G@warning{Citation ‘\@citeb’ undefined}{}}
50 }%
```

But before `\begin{document}` we raised an error message in the past but as of 2021/05 not any longer.

```
51 \% \G@warning{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.

```
52 \AddToHook{begindocument/end}[kernel]{\nocite{#1}%
53 }%
54 \esphack
55 {/2ekernel | latexrelease}
56 {latexrelease}\EndIncludeInRelease
57 {latexrelease}\IncludeInRelease{0000/00/00}%
58 {latexrelease} {\nocite}{Allow nocite in preamble}%
59 {latexrelease}
60 {latexrelease}\def\nocite#1{\esphack
61 {latexrelease}\ifx\@onlypreamble\document
62 {latexrelease}\@for\@citeb:=#1\do{%
63 {latexrelease}\edef\@citeb{\expandafter\@firstofone\@citeb}%
64 {latexrelease}\if@filesw\immediate\write\@auxout{\string\citation{\@citeb}}\fi
65 {latexrelease}\@ifundefined{b@\@citeb}{\G@refundefinedtrue
66 {latexrelease}\G@warning{Citation ‘\@citeb’ undefined}{}}
67 {latexrelease}\else
68 {latexrelease}\G@warning{Cannot be used in preamble}\@eha
69 {latexrelease}\fi
70 {latexrelease}\esphack
71 {latexrelease}
72 {latexrelease}\EndIncludeInRelease
73 {*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 BiBTEX in the first place).

74 \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`

75 \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.

76 \let\@cite@ofmt\hbox

(*End definition for \@cite@ofmt.*)

`\@biblabel`

77 \def\@biblabel#1{\#1}

78 \ekernel

(*End definition for \@biblabel.*)

# File R

## 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 \emptokena \expandafter{\@themark}%
27 \mark{\the\emptokena}%
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 \emptokena \expandafter{\@themark}%
35 \mark{\the\emptokena}%
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 A sloppypar environment is equivalent to {\par \sloppy ... \par}.
 62 \def\sloppypar{\par\sloppy}
 63 \def\endsloppypar{\par}

\fussy Resets TEX'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 LATEX default is no overfull box rule. Changed by document class option.
 69 \overfullrule Opt

(End definition for \overfullrule.)

 70 ⟨/2ekernel⟩

```

# File S

## 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.

**filecontents**

Note that class files have the extension `.cls`, packages have the extension `.sty`.

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.

Similar to \RequirePackage, but for classes, may not be used in package files.

Packages can pass options to other packages using:

\PassOptionsToPackage{\options}{\package}.

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.

To find out if a package has already been loaded, use

```
\IfPackageLoadedTF{(package)}{true}{false}
```

or the old name `\@ifpackageloaded`.

To find out if a package has already been loaded with a version equal to or more recent than `(date)`, use

```
\IfPackageAtLeastTF{(package)}{(date)}{true}{false}
```

or the old name `\@ifpackagelater`.

To test the format date use

```
\IfFormatAtLeastTF{(date)}{true}{false}
```

To find out if a package has already been loaded with at least the options `<options>`,

use

```
\IfPackageLoadedWithOptionsTF{(package)}{options}{true}{false}
```

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.

To define a builtin option, use `\DeclareOption{<name>}{<code>}`.

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@pkgextension.)

```
\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 ⟨latexrelease⟩ \ifnum\expandafter\@parse@version#1//00@nil<%
202 ⟨latexrelease⟩ \expandafter\@parse@version#2//00@nil
203 ⟨latexrelease⟩ \expandafter\@secondoftwo
204 ⟨latexrelease⟩ \else
205 ⟨latexrelease⟩ \expandafter\@firstoftwo
206 ⟨latexrelease⟩ \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{\@if@options\@pkgextension}
220 \def\@ifclasswith{\@if@options\@clsextension}
```

```

221 \def\@ifoptions#1#2{%
222 \@expandtwoargs\@ifoptions{\@optionlist{#2.#1}}}
223 //ekernel
224 <latexrelease>\IncludeInRelease{2017/01/01}%
225 <latexrelease> {\@ifoptions{Spaces in option clash check}%
226 {*2ekernel | latexrelease}%
227 \def\@ifoptions#1#2{%
228 \let\reserved@a\@firstoftwo
229 \edef\reserved@b{\zap@space#2 \empty}%
230 \for\reserved@b:=\reserved@b\do{%
231 \ifx\reserved@b\empty
232 \else
233 \expandafter\in@\expandafter{\expandafter,\reserved@b,}{,#1,}%
234 \ifin@
235 \else
236 \let\reserved@a\@secondoftwo
237 \fi
238 \fi
239 }%
240 \reserved@a}
241 //ekernel | latexrelease)
242 <latexrelease>\EndIncludeInRelease
243 <latexrelease>\IncludeInRelease{0000/00/00}%
244 <latexrelease> {\@ifoptions{Spaces in option clash check}%
245 <latexrelease>\def\@ifoptions#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 \fi
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 //ekernel
262 {*2ekernel | latexrelease)
263 <latexrelease>\IncludeInRelease{2021/11/15}%
264 <latexrelease> {\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 <texrelease>\IncludeInRelease{0000/00/00}%
360 <texrelease> {\ProvidesClass}{Track \ProvidesPackage}%
361 {*2ekernel | latexrelease}
362 \let\ProvidesClass\ProvidesPackage
363 \onlypreamble\ProvidesClass
364 {/2ekernel | latexrelease}
365 <texrelease>\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 <texrelease>\IncludeInRelease{2021/06/01}%
380 <texrelease> {\@passoptions}{Raw option lists}%
381 {*2ekernel | latexrelease}
382 \def\@passoptions#1#2#3{%
383 \expl@@@filehook@set@curr@file@nNN
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\g@addto@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\* 433 \def\DeclareOption{%

```

434 \let\@fileswith@ptions\@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.)

\@useoption Execute the code for the current option.

```

565 〈/2ekernel〉
566 〈\latexrelease〉\IncludeInRelease{2021/06/01}%
567 〈\latexrelease〉 {\@useoption}{filter unused option list}%
568 〈*2ekernel | latexrelease〉
569 \def\@useoption{%
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〉 {\@useoption}{filter unused option list}%
578 〈\latexrelease〉\def\@useoption{%
579 〈\latexrelease〉 \@expandtwoargs\@removeelement\CurrentOption
580 〈\latexrelease〉 \unusedoptionlist\unusedoptionlist
581 〈\latexrelease〉 \csname ds@\CurrentOption\endcsname}
582 〈\latexrelease〉\EndIncludeInRelease
583 〈*2ekernel〉
584 \@onlypreamble\@useoption
```

(End definition for \@useoption.)

\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\@pkextension}
620 \onlypreamble\RequirePackage

```

(*End definition for \RequirePackage.*)

\LoadClass Load class.

```

621 \def\LoadClass{%
622 \ifx\@currext\@pkextension
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 \expandafter\let\csname opt@#3.#1\expandafter\endcsname
645 \csname opt@\currname.\currname\endcsname
646 #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 `\@nnnil` 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\@nnnil##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 \end{IncludeInRelease}
751 \IncludeInRelease{2017/01/01}%
752 {\@fileswith@pti@ns}{ifx tests in \@fileswith@pti@ns}%
753 \def{@fileswith@pti@ns#1[#2]#3[#4]}%
754 \ifx\@clsextension
755 \ifx\@classoptionslist\relax
756 \xdef\@classoptionslist{\zap@space#2 \empty}%
757 \def\reserved@a{%
758 \@onefilewithoptions#3[{\#2}][{\#4}]#1%
759 \documentclasshook}%
760 \else
761 \def\reserved@a{%
762 \@onefilewithoptions#3[{\#2}][{\#4}]#1}%
763 \fi
764 \else
765 \def\reserved@b##1{%
766 \ifx\@nnil##1\relax\else
767 \ifx\@nnil##1\@nnil\else
768 \noexpand\@onefilewithoptions##1[{\#2}][{#4}]%
769 \noexpand\@pkgextension
770 \fi
771 \expandafter\reserved@b
772 \fi}%
773 \edef\reserved@a{\zap@space#3 \empty}%
774 \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nnil,}%
775 \fi
776 \reserved@a}
777 \end{IncludeInRelease}
778 \IncludeInRelease{0000/00/00}%
779 {\@fileswith@pti@ns}{ifx tests in \@fileswith@pti@ns}%
780 \def{@fileswith@pti@ns#1[#2]#3[#4]}%
781 \ifx\@clsextension
782 \ifx\@classoptionslist\relax
783 \xdef\@classoptionslist{\zap@space#2 \empty}%
784 \def\reserved@a{%
785 \@onefilewithoptions#3[{\#2}][{\#4}]#1%
786 \documentclasshook}%
787 \else
788 \def\reserved@a{%
789 \@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.

\load@onefilewithoptions

```

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
```

If the package is already loaded, check that there were no option clashes:

```
823 {\@ifoptions\@currext{\@currname}{#2}{}}%
824 {\@latex@error
825 {Option clash for \@cls@pkg\space \@currname}%
826 {The package \@currname\space has already been loaded
827 with options:\MessageBreak
828 \space\space[\@optionlist{\@currname.\@currext}]\MessageBreak
829 There has now been an attempt to load it
830 with options\MessageBreak
831 \space\space[#2]\MessageBreak
832 Adding the global options:\MessageBreak
833 \space\space
834 \@optionlist{\@currname.\@currext},#2\MessageBreak
835 to your \noexpand\documentclass declaration may fix this.%\MessageBreak
836 Try typing \space <return> \space to proceed.}}%
837 {\@firstofone}%
838 {\@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!

```
840 \@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 `\@missingfile@error`. For checking if the file exists we use `\@currpath` (usually empty) before `\@currname`.

```
841 \@IfFileExists{\@currpath\@currname.\@currext}{}%
842 {\@missing@oneline@withoptions{#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.

```
843 \@ifx\@currname\@empty
844 \@expandafter\@gobble
845 \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@oneline@withoptions`. `\set@curr@file` also needs the file path.

```
846 \@disable@packageload@do{\@currname.\@currext}%
847 {\@expl@@@filehook@file@push@@
848 \set@curr@file{\@currpath\@currname.\@currext}%
849 \@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.

```

850 \expandafter\@swaptwoargs\expandafter
851 {\expandafter{\@currpkg@reqd}}%
852 {%
853 \edef\@currpkg@reqd{\@currname.\@currext}%
854 \ifx\CurrentFile\CurrentFileUsed
855 \else
856 \filename@parse\@curr@file
857 \edef\@currpath{\string@makeletter\filename@area}%
858 \edef\@currname{\string@makeletter\filename@base}%
859 \edef\@currext{\string@makeletter\filename@ext}%
860 \fi
861 \load@onefile@withoptions{#2}%
862 \def\@currpkg@reqd{\@currpkg@reqd}%
863 }%

```

Now just clean up and exit.

```

864 \expl@@@filehook@file@pop@@}%
865 \expandafter\@firstofone
866 \fi}%

```

Except in the case where `\@currname` is empty, the date is checked against the date marked in the package file:

```

867 {\@if@ter\@currext{\@currname}{#3}{}}%
868 {\@latex@warning@no@line
869 {You have requested,\on@line,
870 version\MessageBreak
871 '#3' of \cls@pkg\space \currname,\MessageBreak
872 but only version\MessageBreak
873 '\csname ver@\currname.\@currext\endcsname'\MessageBreak
874 is available}}%
875 \ifx\@currext\@clsextension\let\LoadClass@twoloadclasserror\fi}%
876 \popfilename
877 \reset@ptions}
878 \let\@currpkg@reqd\empty
879 \onlypreamble\onefile@withoptions
The kernel no longer uses \unprocessedoptions
880 \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.

```

881 \def\missing@onefile@withoptions#1{%

```

```

882 \@missingfileerror{\@currpath\@currname}\@currext
883 \global\let\@currpath\@missingfile@area
884 \global\let\@currname\@missingfile@base
885 \global\let\@currext\@missingfile@ext}

```

Now the code that actually does the file loading:

```

\load@onefile@withoptions 886 \def\load@onefile@withoptions#1{%
887 \let\CurrentOption\@empty
888 \reset@options

```

Grab everything in a macro, so the parameter stack is popped before any processing begins.

```

889 \def\reserved@a{%
890 \@pass@options\@currext{#1}{\@currname}%
891 \expandafter\let
892 \csname opt@\@currpkg@reqd\expandafter\endcsname
893 \csname opt@\@currname.\@currext\endcsname
894 \expandafter\let
895 \csname @raw@opt@\@currpkg@reqd\expandafter\endcsname
896 \csname @raw@opt@\@currname.\@currext\endcsname
897 \global\expandafter
898 \let\csname ver@\@currname.\@currext\endcsname\@empty

```

We initialize `\...-h@k` here and only if we load the file so that it remains undefined otherwise.

```
899 \expandafter\let\csname\@currname.\@currext-h@k\endcsname\@empty
```

When the current extension is `\@pkgextension` 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.

```

900 %-----
901 \ifx\@currext\@pkgextension
902 \UseHook{package/before}%
903 \UseOneTimeHook{package/\@currname/before}%
904 \else
905 \ifx\@currext\@clsextension
906 \UseHook{class/before}%
907 \UseOneTimeHook{class/\@currname/before}%
908 \fi
909 \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.

```

910 \InputIfFileExists{\@currpath\@currpkg@reqd}{}%
911 {\@latex@error
912 {The \@cls@pkg\space\@currpkg@reqd\space failed to load}\@ehd}%
913 %-----

```

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.

```

914 \expandafter\let\csname unprocessedoptions-\@currname.\@currext\endcsname
915 \@@unprocessedoptions
916 \csname \@currname.\@currext-h@@k\endcsname
917 \expandafter\let\csname \@currname.\@currext-h@@k\endcsname
918 \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.

```

919 \ifx\@unprocessedoptions\relax
920 \let\@unprocessedoptions\undefined

```

Otherwise run the per package set of unused options.

```

921 \else
922 \csname unprocessedoptions-\@currname.\@currext\endcsname
923 \fi

```

In either case we drop the macro afterwards as it is no longer needed.

```

924 \expandafter\let
925 \csname unprocessedoptions-\@currname.\@currext\endcsname
926 \undefined

```

And same procedure, James, when we are finished loading, except that the hook order is now reversed.

```

927 %-----
928 \ifx\@currext\@pkgextension
929 \UseOneTimeHook{package/\@currname/after}%
930 \UseHook{package/after}%
931 \else
932 \ifx\@currext\@clsextension
933 \UseOneTimeHook{class/\@currname/after}%
934 \UseHook{class/after}%
935 \fi
936 \fi}%
937 %-----
938 \@ifl@aded\@currext\@currname{}{\reserved@a}%

```

Now declare the non-generic package and class hooks used above:

```

939 \NewHook{package/before}
940 \NewHook{class/before}
941 \NewReversedHook{package/after}
942 \NewReversedHook{class/after}

943 </2ekernel | latexrelease>
944 <latexrelease>\EndIncludeInRelease
945 <latexrelease>\IncludeInRelease{0000/00/00}%
946 <latexrelease> {\@onefilewithoptions}{Hooks and unused options issue}%
947 <latexrelease>

```

Because of the way \onefilewithoptions is changed for rollback handling below we have to define \load@onefilewithoptions when rolling back!

```

948 <latexrelease>\def\load@onefilewithoptions#1[#2] [#3]#4{%
949 <latexrelease> \pushfilename
950 <latexrelease> \xdef\@currname{#1}%
951 <latexrelease> \global\let\@currext#4%
952 <latexrelease> \let\CurrentOption\empty
953 <latexrelease> \reset@options

```

```

954 <|latexrelease> \makeatletter
955 <|latexrelease> \def\reserved@a{%
956 <|latexrelease> \@if@aded\@currext{#1}%
957 <|latexrelease> {\@if@options\@currext{#1}{#2}{}}%
958 <|latexrelease> {\@latex@error
959 <|latexrelease> {Option clash for \@cls@pkg\space #1}%
960 <|latexrelease> {The package #1 has already been loaded
961 <|latexrelease> with options:\MessageBreak
962 <|latexrelease> \space\space[\@optionlist{#1.\@currext}]\MessageBreak
963 <|latexrelease> There has now been an attempt to load it
964 <|latexrelease> with options\MessageBreak
965 <|latexrelease> \space\space[#2]\MessageBreak
966 <|latexrelease> Adding the global options:\MessageBreak
967 <|latexrelease> \space\space
968 <|latexrelease> \@optionlist{#1.\@currext},#2\MessageBreak
969 <|latexrelease> to your \noexpand\documentclass declaration may fix this.%\MessageBreak
970 <|latexrelease> Try typing \space <return> \space to proceed.}}%
971 <|latexrelease> {\@pass@ptions\@currext{#2}{#1}%
972 <|latexrelease> \global\expandafter
973 <|latexrelease> \let\csname ver@\@currname.\@currext\endcsname\@empty
974 <|latexrelease> \expandafter\let\csname\@currname.\@currext-h@k\endcsname\@empty
975 <|latexrelease> \InputIfFileExists
976 <|latexrelease> {\@currname.\@currext}%
977 <|latexrelease> {}%
978 <|latexrelease> {\@missingfileerror\@currname\@currext}%
979 <|latexrelease> \let\@unprocessedoptions\@unprocessedoptions
980 <|latexrelease> \csname\@currname.\@currext-h@k\endcsname
981 <|latexrelease> \expandafter\let\csname\@currname.\@currext-h@k\endcsname
982 <|latexrelease> \undefined
983 <|latexrelease> \@unprocessedoptions}%
984 <|latexrelease> \@if@ter\@currext{#1}{#3}{}}%
985 <|latexrelease> {\@latex@warning@no@line
986 <|latexrelease> {You have requested,\on@line,
987 <|latexrelease> version\MessageBreak
988 <|latexrelease> '#3' of \@cls@pkg\space #1,\MessageBreak
989 <|latexrelease> but only version\MessageBreak
990 <|latexrelease> '\csname ver@\#1.\@currext\endcsname'\MessageBreak
991 <|latexrelease> is available}}}%
992 <|latexrelease>
993 <|latexrelease> \ifx\@currext\@clsextension\let\LoadClass\@twoloadclasserror\fi
994 <|latexrelease> \popfilename
995 <|latexrelease> \reset@ptions}%
996 <|latexrelease> \reserved@a}
997 <|latexrelease>
998 <|latexrelease> \let\load@onefile@withoptions \undefined
999 <|latexrelease> \let\@missing@onefile@withoptions \undefined
1000 <|latexrelease>
1001 <|latexrelease> \EndIncludeInRelease
1002 <|2ekernel>

(End definition for \@files@with@ptions and others.)

```

\@files@with@ptions Save the definition (for error checking).

```

1003 \let\@files@with@ptions\@files@with@ptions
1004 \onlypreamble\@files@with@ptions

```

(End definition for \@@files with @ptions.)

\@reset@ptions Reset the default option, and clear lists of declared options.

```
1005 \def\@reset@ptions{%
1006 \global\ifx\@currext\@clsextension
1007 \let\default@ds\OptionNotUsed
1008 \else
1009 \let\default@ds\@unknownoptionerror
1010 \fi
1011 \global\let\ds@\emptyset
1012 \global\let\@declaredoptions\emptyset}
1013 \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.

```
1014 \ifx\@begindocumenthook\undefined
1015 \let\@begindocumenthook\emptyset
1016 \fi
1017 \let\@enddocumenthook\emptyset
```

(End definition for \@begindocumenthook and \@enddocumenthook.)

\AtEndOfPackage The access functions.

```
1018 \def\AtEndOfPackage{%
1019 \expandafter\g@addto@macro\csname\@currname.\@currext-h@@k\endcsname}
1020 \let\AtEndOfClass\AtEndOfPackage
1021 \onlypreamble\AtEndOfPackage
1022 \onlypreamble\AtEndOfClass

1023 </2ekernel>
1024 <*2ekernel | latexrelease>
1025 <latexrelease>\IncludeInRelease{2020/10/01}%
1026 <latexrelease> {\AtBeginDocument}{Use hook system}%
1027 \DeclareRobustCommand\AtBeginDocument{\AddToHook{begindocument}}
1028 \DeclareRobustCommand\AtEndDocument {\AddToHook{enddocument}}
1029 %\DeclareRobustCommand\AtEndDocument {\AddToHook{env/document/end}} % alternative impl
1030 </2ekernel | latexrelease>
1031 <latexrelease>\EndIncludeInRelease
1032 <latexrelease>\IncludeInRelease{0000/00/00}%
1033 <latexrelease> {\AtBeginDocument}{Use hook system}%
1034 <latexrelease>
1035 <latexrelease>\DeclareRobustCommand\AtBeginDocument{\g@addto@macro\@begindocumenthook}
1036 <latexrelease>\DeclareRobustCommand\AtEndDocument{\g@addto@macro\@enddocumenthook}
1037 <latexrelease>
1038 <latexrelease>\EndIncludeInRelease
1039 <*2ekernel>
```

```

1040 \Qonlypreamble\AtBeginDocument
(End definition for \AtEndOfPackage and others.)

\@cls@pkg The current file type.
1041 \def\@cls@pkg{%
1042 \ifx\@currext\@clsextension
1043 document class%
1044 \else
1045 package%
1046 \fi}
1047 \Qonlypreamble\@cls@pkg

(End definition for \@cls@pkg.)

\@unknownoptionerror Bad option.
1048 \def\@unknownoptionerror{%
1049 \Qlatex@error
1050 {Unknown option ‘\CurrentOption’ for \@cls@pkg\space‘\currname’}%
1051 {The option ‘\CurrentOption’ was not declared in
1052 \@cls@pkg\space‘\currname’, perhaps you\MessageBreak
1053 misspelled its name.
1054 Try typing \space <return>
1055 \space to proceed.}}
1056 \Qonlypreamble\@unknownoptionerror

(End definition for \@unknownoptionerror.)

\@unprocessedoptions Declare an error for each option, unless a \ProcessOptions occurred.
1057 \def\@unprocessedoptions{%
1058 \ifx\@currext\@pkgextension
1059 \edef\@curroptions{\@optionlist{\currname.\@currext}}%
1060 \Qfor\CurrentOption:=\@curroptions\do{%
1061 \ifx\CurrentOption\empty\else\@unknownoptionerror\fi}%
1062 \fi}
1063 \Qonlypreamble\@unprocessedoptions
1064 \Qonlypreamble\@unprocessedoptions

(End definition for \@unprocessedoptions.)

\@badrequireerror \RequirePackage or \LoadClass occurs in the options section.
1065 \def\@badrequireerror#1[#2]#3[#4]{%
1066 \Qlatex@error
1067 {\noexpand\RequirePackage or \noexpand\LoadClass
1068 in Options Section}%
1069 {The \@cls@pkg\space ‘\currname’ is defective.\MessageBreak
1070 It attempts to load ‘#3’ in the options section, i.e.,\MessageBreak
1071 between \noexpand\DeclareOption and \string\ProcessOptions.}}
1072 \Qonlypreamble\@badrequireerror

(End definition for \@badrequireerror.)

```

```

\@two@loadclass@error Two \LoadClass in a class.
1073 \def\@two@loadclass@error{%
1074 \@latex@error
1075 {Two \noexpand\LoadClass commands}%
1076 {You may only use one \noexpand\LoadClass in a class file}%
1077 \only@preamble\@two@loadclass@error
(End definition for \@two@loadclass@error.)}

\@twoclasses@error Two \documentclass or \documentstyle.
1078 \def\@twoclasses@error#1{%
1079 \@latex@error
1080 {Two \noexpand\documentclass or \noexpand\documentstyle commands}%
1081 {The document may only declare one class.}\@gobble}
1082 \only@preamble\@twoclasses@error
(End definition for \@twoclasses@error.)

```

## 4.2 Providing shipment

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

```

1083 \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 into the file.

```

1084 </2ekernel>
1085 <*2ekernel | latexrelease>
1086 <latexrelease>\IncludeInRelease{2020/10/01}%
1087 <latexrelease> {\filec@ntents}{\Define \q@curr@file directly (gh/220)}%
1088 %

```

We use @tempswa to mean no preamble writing and reuse @files w to indicate no overwriting:

```

1089 \def\filecontents{\@tempswatru@filestrue
1090 \@ifnextchar[\filec@ntents@opt\filec@ntents
1091 }
1092 \namedef{filecontents*}{\@tempswafalse\@filestrue
1093 \@ifnextchar[\filec@ntents@opt\filec@ntents
1094 }

```

To handle the optional argument we execute for each option the command \filec@ntents@OPTION if it exist or complain about unknown option.

```

1095 \def\filec@ntents@opt[#1]{%
1096 \edef\@fortmp{\zap@space#1 \empty}%
1097 \@for\reserved@a:=\@fortmp\do{%
1098 \ifcsname filec@ntents@\reserved@a\endcsname
1099 \csname filec@ntents@\reserved@a\endcsname
1100 \else
1101 \@latex@error{Unknown filecontents option \reserved@a}%
1102 {Valid options are force (or overwrite), nosearch, noheader}%
1103 \fi}%
1104 \filec@ntents
1105 }

```

Option `force` (or `overwrite`) changes the overwriting switch

```
1106 \let\filec@ntents@force@\fileswfalse
1107 \let\filec@ntents@overwrite@\fileswfalse % alternative name
```

and option `noheader` the preamble switch (which is equivalent to using the star form of the environment).

```
1108 \let\filec@ntents@noheader@\tempswfalse
```

Option `nosearch` only checks the current directory not the whole TeX tree for the existence of the file to write.

```
1109 \def\filec@ntents@nosearch{
1110 \let\filec@ntents@checkdir@\currdir
1111 \def\filec@ntents@where{in current directory}}
```

By default we search the whole tree:

```
1112 \let\filec@ntents@checkdir@\empty
1113 \def\filec@ntents@where{exists on the system}
1114 \begingroup%
1115 \tempcnta=1
1116 \loop
1117 \catcode\@tempcnta=12 %
1118 \advance\@tempcnta\one %
1119 \ifnum\@tempcnta<32 %
1120 \repeat
1121 \catcode`*=11 %
1122 \catcode`\^M\active%
1123 \catcode`\^L\active\let\^L\relax%
1124 \catcode`\^I\active%
1125 \gdef\filec@ntents#1%
1126 \set@curr@file{\filec@ntents@checkdir#1}%
1127 \edef\q@curr@file{"\curr@file"}%
```

LuaTeX has more writes (and 18 is safe here).

```
1128 \chardef\reserved@c\ifx\directlua\undefined 15 \else 127 \fi%
1129 \openin\@inputcheck\q@curr@file \space %
1130 \ifeof\@inputcheck%
1131 \at@note@no@line%
 {Writing file '\currdir\curr@file'}%
1132
1133 \ch@ck7\reserved@c\write\relax%
1134 \immediate\openout\reserved@c\q@curr@file\relax%
1135 \else%
1136 \if@filesw%
 \at@note@no@line%
 {File '\curr@file' already \filec@ntents@where.\MessageBreak%
 Not generating it from this source}%">
1137 \let\write@gobbletwo%
 \let\closeout\gobble%
1138 \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!

```

1143 \edef\reserved@b{\detokenize\expandafter{\jobname}}%
1144 \ifx@\curr@file\reserved@b%
1145 \Qfilestrue%
1146 \else%
1147 \edef\reserved@b{\reserved@b\detokenize{.tex}}%
1148 \ifx@\curr@file\reserved@b%
1149 \Qfilestrue%
1150 \fi%
1151 \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.

```

1152 \ch@ck7\reserved@c\write\relax%
1153 \if@files w% % Foul ... trying to overwrite \jobname!
1154 \Qlatex@error{Trying to overwrite '\jobname.tex'}{You can't %
1155 write to the file you are reading from!\MessageBreak%
1156 Data is written to screen instead.}%
1157 \else%
1158 \Qlatex@warning{no@line}%
1159 {Writing or overwriting file '@currdir@\curr@file'}%
1160 \immediate\openout\reserved@c\q@curr@file\relax%
1161 \fi%
1162 \fi%
1163 \fi%

```

Closing the `\@inputcheck` is done here to avoid having to do this in each branch.

```

1164 \closein\@inputcheck%
1165 \if@tempswa%
1166 \immediate\write\reserved@c{%
1167 \Qpercentchar\Qpercentchar\space%
1168 \expandafter\Qgobble\string\LaTeXe file '@curr@file'^^J%
1169 \Qpercentchar\Qpercentchar\space generated by the %
1170 '@currenvir' \expandafter\Qgobblefour\string\newenvironment^^J%
1171 \Qpercentchar\Qpercentchar\space from source '\jobname' on %
1172 \number\year/\two@digits\month/\two@digits\day.^^J%
1173 \Qpercentchar\Qpercentchar}%
1174 \fi%
1175 \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.

```

1176 \count@ 128\relax%
1177 \loop%
1178 \catcode\count@ 11\relax%
1179 \advance\count@ \one%
1180 \ifnum\count@<\@ccclvi%
1181 \repeat%
1182
1183 \edef\E{\@backslashchar end\string{\@currenvir\string}{}%
1184 \edef\reserved@b{%
1185 \def\noexpand\reserved@b{%

```

```

1185 #####1\E#####2\E#####3\relax}%
1186 \reserved@b{%
1187 \ifx\relax##3\relax%
There was no \end{filecontents}
1188 \immediate\write\reserved@c{##1}%
1189 \else%
There was a \end{filecontents}, so stop this time.
1190 \edef^~M{\noexpand\end{@currenvir}}%
1191 \ifx\relax##1\relax%
1192 \else%
Text before the \end, write it with a warning.
1193 @latex@warning{Writing text ‘##1’ before %
1194 \string\end{@currenvir}\MessageBreak
1195 as last line of \@curr@file}%
1196 \immediate\write\reserved@c{##1}%
1197 \fi%
1198 \ifx\relax##2\relax%
1199 \else%
Text after the \end, ignore it with a warning.
1200 @latex@warning{%
1201 Ignoring text ‘##2’ after \string\end{@currenvir}}%
1202 \fi%
1203 \fi%
1204 ^~M}%
1205 \catcode`^~L\active%
1206 \let\L\@undefined%
1207 \def^~L{\expandafter\ifx\csname L\endcsname\relax\fi ^~J^~J}%
1208 \catcode`^~I\active%
1209 \let\I\@undefined%
1210 \def^~I{\expandafter\ifx\csname I\endcsname\relax\fi\space}%
1211 \catcode`^~M\active%
1212 \edef^~M##1^~M{%
1213 \noexpand\reserved@b##1\relax}%
1214 \endgroup%
1215 //2ekernel | latexrelease)
1216 <latexrelease>\EndIncludeInRelease
1217 <latexrelease>\IncludeInRelease{2019/10/01}%
1218 <latexrelease> {\filec@ntents}{Spaces in file names + optional arg}%
1219 <latexrelease>
1220 <latexrelease>\def\filecontents{@tempswattrue@files wtrue
1221 <latexrelease> @ifnextchar[\filec@ntents@opt\filec@ntents
1222 <latexrelease>
1223 <latexrelease>\@namedef{filecontents*}{@tempswafalse@files wtrue
1224 <latexrelease> @ifnextchar[\filec@ntents@opt\filec@ntents
1225 <latexrelease>
1226 <latexrelease>\def\filec@ntents@opt [#1]{%
1227 <latexrelease> \edef@fortmp{\zap@space#1 \@empty}%
1228 <latexrelease> @for\reserved@a:=@fortmp\do{%
1229 <latexrelease> \ifcsname filec@ntents@\reserved@a\endcsname
1230 <latexrelease> \csname filec@ntents@\reserved@a\endcsname

```

```

1231 <{latexrelease}> \else
1232 <{latexrelease}> \@latex@error{Unknown filecontents option \reserved@a}%
1233 <{latexrelease}> {Valid options are force (or overwrite), nosearch, noheader}%
1234 <{latexrelease}> \fi}%
1235 <{latexrelease}> \filec@ntents
1236 <{latexrelease}> }
1237 <{latexrelease}> \let\filec@ntents@force\@fileswfalse
1238 <{latexrelease}> \let\filec@ntents@overwrite\@fileswfalse % alternative name
1239 <{latexrelease}> \let\filec@ntents@noheader\@tempswafalse
1240 <{latexrelease}> \def\filec@ntents@nosearch{%
1241 <{latexrelease}> \let\filec@ntents@checkdir\@currdir
1242 <{latexrelease}> \def\filec@ntents@where{in current directory}}
1243 <{latexrelease}> \let\filec@ntents@checkdir\@empty
1244 <{latexrelease}> \def\filec@ntents@where{exists on the system}
1245 <{latexrelease}> \begin{group}%
1246 <{latexrelease}> \tempcnta=1
1247 <{latexrelease}> \loop
1248 <{latexrelease}> \catcode\@tempcnta=12 %
1249 <{latexrelease}> \advance\@tempcnta\@ne %
1250 <{latexrelease}> \ifnum\@tempcnta<32 %
1251 <{latexrelease}> \repeat %
1252 <{latexrelease}> \catcode`*=11 %
1253 <{latexrelease}> \catcode`\~\active%
1254 <{latexrelease}> \catcode`\^L\active\let`^L\relax%
1255 <{latexrelease}> \catcode`\^I\active%
1256 <{latexrelease}> \gdef\filec@ntents#1{%
1257 <{latexrelease}> \set@curr@file{\filec@ntents@checkdir#1}%
1258 <{latexrelease}> \edef\q@curr@file{\expandafter\quote@name\expandafter{\@curr@file}}%
1259 <{latexrelease}> \chardef\reserved@c\ifx\directlua\@undefined 15 \else 127 \fi%
1260 <{latexrelease}> \openin\@inputcheck\q@curr@file \space %
1261 <{latexrelease}> \ifeof\@inputcheck%
1262 <{latexrelease}> \@latex@warning{no@line}%
1263 <{latexrelease}> {Writing file `@\currdir@\curr@file'}%
1264 <{latexrelease}> \ch@ck7\reserved@c\write\relax%
1265 <{latexrelease}> \immediate\openout\reserved@c\q@curr@file\relax%
1266 <{latexrelease}> \else%
1267 <{latexrelease}> \if@files w%
1268 <{latexrelease}> \@latex@warning{no@line}%
1269 <{latexrelease}> {File `@\curr@file' already \filec@ntents@where.\MessageBreak% %
1270 <{latexrelease}> Not generating it from this source}%
1271 <{latexrelease}> \let\write@\gobbletwo%
1272 <{latexrelease}> \let\closeout\gobble%
1273 <{latexrelease}> \else%
1274 <{latexrelease}> \edef\reserved@a{\#1}%
1275 <{latexrelease}> \edef\reserved@a{\detokenize\expandafter{\reserved@a}}%
1276 <{latexrelease}> \edef\reserved@b{\detokenize\expandafter{\jobname}}%
1277 <{latexrelease}> \ifx\reserved@a\reserved@b%
1278 <{latexrelease}> \@files wtrue%
1279 <{latexrelease}> \else%
1280 <{latexrelease}> \edef\reserved@b{\reserved@b\detokenize{.tex}}%
1281 <{latexrelease}> \ifx\reserved@a\reserved@b
1282 <{latexrelease}> \@files wtrue%
1283 <{latexrelease}> \fi%
1284 <{latexrelease}> \fi%

```

```

1285 〈latexrelease〉 \ch@ck7\reserved@c\write\relax%
1286 〈latexrelease〉 \if@filesw% % Foul ... trying to overwrite \jobname!
1287 〈latexrelease〉 \@latex@error{Trying to overwrite ‘\jobname.tex’}{You can’t %
1288 〈latexrelease〉 write to the file you are reading from!}\MessageBreak%
1289 〈latexrelease〉 Data is written to screen instead.}%
1290 〈latexrelease〉 \else%
1291 〈latexrelease〉 \@latex@warning@no@line%
1292 〈latexrelease〉 {Writing or overwriting file ‘\@currdir\@curr@file’}%
1293 〈latexrelease〉 \immediate\openout\reserved@c\q@curr@file\relax%
1294 〈latexrelease〉 \fi%
1295 〈latexrelease〉 \fi%
1296 〈latexrelease〉 \fi%
1297 〈latexrelease〉 \closein\@inputcheck%
1298 〈latexrelease〉 \if@tempswa%
1299 〈latexrelease〉 \immediate\write\reserved@c{%
1300 〈latexrelease〉 \@percentchar\@percentchar\space%
1301 〈latexrelease〉 \expandafter\@gobble\string\LaTeXe file ‘\@curr@file’^J%
1302 〈latexrelease〉 \@percentchar\@percentchar\space generated by the %
1303 〈latexrelease〉 ‘\@currenvir’ \expandafter\@gobblefour\string\newenvironment^J%
1304 〈latexrelease〉 \@percentchar\@percentchar\space from source ‘\jobname’ on %
1305 〈latexrelease〉 \number\year/\two@digits\month/\two@digits\day.^J%
1306 〈latexrelease〉 \@percentchar\@percentchar}%
1307 〈latexrelease〉 \fi%
1308 〈latexrelease〉 \let\do\@makeother\dospecials%
1309 〈latexrelease〉 \count@ 128\relax%
1310 〈latexrelease〉 \loop%
1311 〈latexrelease〉 \catcode\count@ 11\relax%
1312 〈latexrelease〉 \advance\count@ \@ne%
1313 〈latexrelease〉 \ifnum\count@<@\cclvi%
1314 〈latexrelease〉 \repeat%
1315 〈latexrelease〉 \edef\E{\@backslashchar end\string\{@currenvir\string\}}%
1316 〈latexrelease〉 \edef\reserved@b{%
1317 〈latexrelease〉 \def\noexpand\reserved@b{%
1318 〈latexrelease〉 #####1\E#####2\E#####3\relax}%
1319 〈latexrelease〉 \reserved@b{%
1320 〈latexrelease〉 \ifx\relax##3\relax%
1321 〈latexrelease〉 \immediate\write\reserved@c{##1}%
1322 〈latexrelease〉 \else%
1323 〈latexrelease〉 \edef\@M{\noexpand\end\string\{@currenvir}}%
1324 〈latexrelease〉 \ifx\relax##1\relax%
1325 〈latexrelease〉 \else%
1326 〈latexrelease〉 \@latex@warning{Writing text ‘##1’ before %
1327 〈latexrelease〉 \string\end\string\{@currenvir}\MessageBreak as last line of \@curr@file}%
1328 〈latexrelease〉 \immediate\write\reserved@c{##1}%
1329 〈latexrelease〉 \fi%
1330 〈latexrelease〉 \ifx\relax##2\relax%
1331 〈latexrelease〉 \else%
1332 〈latexrelease〉 \@latex@warning{%
1333 〈latexrelease〉 Ignoring text ‘##2’ after \string\end\string\{@currenvir}}%
1334 〈latexrelease〉 \fi%
1335 〈latexrelease〉 \fi%
1336 〈latexrelease〉 ^M}%
1337 〈latexrelease〉 \catcode‘\^\L\active%
1338 〈latexrelease〉 \let\L\undefined%

```

```

1339 <|latexrelease> \def^~L{|\expandafter\ifx\csname L\endcsname\relax\fi ^~J^~J}%
1340 <|latexrelease> \catcode`^~I\active%
1341 <|latexrelease> \let\I@undefined%
1342 <|latexrelease> \def^~I{|\expandafter\ifx\csname I\endcsname\relax\fi\space}%
1343 <|latexrelease> \catcode`^~M\active%
1344 <|latexrelease> \edef^~M#1^~M{%
1345 <|latexrelease> \noexpand\reserved@b##1\relax}%
1346 <|latexrelease>\endgroup%
1347 <|latexrelease>\EndIncludeInRelease
1348 <|latexrelease>\IncludeInRelease{0000/00/00}%
1349 <|latexrelease> {\filec@ntents}{Spaces in file names + optional arg}%
1350 <|latexrelease>
1351 <|latexrelease>\let\filec@ntents@opt \undefined
1352 <|latexrelease>\let\filec@ntents@force \undefined
1353 <|latexrelease>\let\filec@ntents@overwrite \undefined
1354 <|latexrelease>\let\filec@ntents@noheader \undefined
1355 <|latexrelease>\let\filec@ntents@nosearch \undefined
1356 <|latexrelease>\let\filec@ntents@checkdir \undefined
1357 <|latexrelease>\let\filec@ntents@where \undefined
1358 <|latexrelease>
1359 <|latexrelease>\begingroup%
1360 <|latexrelease>\@tempcnta=1
1361 <|latexrelease>\loop
1362 <|latexrelease> \catcode`\@tempcnta=12 %
1363 <|latexrelease> \advance\@tempcnta\@ne %
1364 <|latexrelease>\ifnum\@tempcnta<32 %
1365 <|latexrelease>\repeat %
1366 <|latexrelease>\catcode`*=11 %
1367 <|latexrelease>\catcode`^~M\active%
1368 <|latexrelease>\catcode`^~L\active\let^~L\relax%
1369 <|latexrelease>\catcode`^~I\active%
1370 <|latexrelease>
1371 <|latexrelease>\gdef\filec@ntents#1{%
1372 <|latexrelease> \openin\@inputcheck#1 %
1373 <|latexrelease> \ifeof\@inputcheck%
1374 <|latexrelease> \@latex@warning@no@line%
1375 <|latexrelease> {Writing file `'\@currdir#1'}}%
1376 <|latexrelease> \chardef\reserved@c15 %
1377 <|latexrelease> \ch@ck7\reserved@c\write%
1378 <|latexrelease> \immediate\openout\reserved@c#1\relax%
1379 <|latexrelease> \else%
1380 <|latexrelease> \closein\@inputcheck%
1381 <|latexrelease> \@latex@warning@no@line%
1382 <|latexrelease> {File '#1' already exists on the system.\MessageBreak%
1383 <|latexrelease> Not generating it from this source}%
1384 <|latexrelease> \let\write\@gobbletwo%
1385 <|latexrelease> \let\closeout\@gobble%
1386 <|latexrelease> \fi%
1387 <|latexrelease> \if@tempswa%
1388 <|latexrelease> \immediate\write\reserved@c{%
1389 <|latexrelease> \percentchar\percentchar\space%
1390 <|latexrelease> \expandafter\@gobble\string\LaTeXe file '#1'^~J}%
1391 <|latexrelease> \percentchar\percentchar\space generated by the %
1392 <|latexrelease> '@currenvir' \expandafter\@gobblefour\string\newenvironment'^~J}%

```

```

1393 <|latexrelease> \@percentchar\@percentchar\space from source '\jobname' on %
1394 <|latexrelease> \number\year/\two@digits\month/\two@digits\day.^~J%
1395 <|latexrelease> \@percentchar\@percentchar}%
1396 <|latexrelease> \fi%
1397 <|latexrelease> \let\do\@makeother\dospecials%
1398 <|latexrelease> \count@ 128\relax%
1399 <|latexrelease> \loop%
1400 <|latexrelease> \catcode\count@ 11\relax%
1401 <|latexrelease> \advance\count@ \@ne%
1402 <|latexrelease> \ifnum\count@<\@cclvi%
1403 <|latexrelease> \repeat%
1404 <|latexrelease> \edef\E{\@backslashchar end\string{\@currenvir\string}}%
1405 <|latexrelease> \edef\reserved@b{%
1406 <|latexrelease> \def\noexpand\reserved@b{%
1407 <|latexrelease> #####1\E#####2\E#####3\relax}%
1408 <|latexrelease> \reserved@b{%
1409 <|latexrelease> \ifx\relax##3\relax%
1410 <|latexrelease> \immediate\write\reserved@c{##1}%
1411 <|latexrelease> \else%
1412 <|latexrelease> \edef^~M{\noexpand\end{\@currenvir}}%
1413 <|latexrelease> \ifx\relax##1\relax%
1414 <|latexrelease> \else%
1415 <|latexrelease> \@latex@warning{Writing text '##1' before %
1416 <|latexrelease> \string\end{\@currenvir}MessageBreak as last line of #1}%
1417 <|latexrelease> \immediate\write\reserved@c{##1}%
1418 <|latexrelease> \fi%
1419 <|latexrelease> \ifx\relax##2\relax%
1420 <|latexrelease> \else%
1421 <|latexrelease> \@latex@warning{%
1422 <|latexrelease> Ignoring text '##2' after \string\end{\@currenvir}}%
1423 <|latexrelease> \fi%
1424 <|latexrelease> \fi%
1425 <|latexrelease> ^~M}%
1426 <|latexrelease> \catcode`\^~L\active%
1427 <|latexrelease> \let\L\@undefined%
1428 <|latexrelease> \def^~L{\expandafter\ifx\csname L\endcsname\relax\fi ^~J^~J}%
1429 <|latexrelease> \catcode`\^~I\active%
1430 <|latexrelease> \let\I\@undefined%
1431 <|latexrelease> \def^~I{\expandafter\ifx\csname I\endcsname\relax\fi\space}%
1432 <|latexrelease> \catcode`\^~M\active%
1433 <|latexrelease> \edef^~M##1^~M{%
1434 <|latexrelease> \noexpand\reserved@b##1\E\relax}%
1435 <|latexrelease> \endgroup%
1436 <|latexrelease>\EndIncludeInRelease
1437 <|latexrelease>{*2ekernel}
1438
1439 \begingroup
1440 \catcode`|=|\catcode`\%
1441 \catcode`\#=12
1442 \catcode`*=11
1443 \gdef\@percentchar{%
1444 \gdef\endfilecontents{|%
1445 \immediate\closeout\reserved@c
1446 \def\T##1##2##3{|%

```

```

1447 \ifx##1@undefined\else
1448 \@latex@warning@no@line{##2 has been converted to Blank ##3e}|
1449 \fi}|
1450 \T\L{Form Feed}{Lin}|
1451 \T\I{Tab}{Spac}|
1452 \immediate\write\@unused{}}
1453 \global\let\endfilecontents*\endfilecontents

```

We no longer prevent the code to be used after begin document (no rollback needed for this change).

```

1454 %\onlypreamble\filecontents
1455 %\onlypreamble\endfilecontents
1456 %\onlypreamble\filecontents*
1457 %\onlypreamble\endfilecontents*
1458 \endgroup
1459 %\onlypreamble\filecontents
(End definition for \filecontents and \endfilecontents.)

```

## 5 Package/class rollback mechanism

```

1460 </2ekernel>
1461 <*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.

```

1462 <*tracerollback>
1463 %\let\pkgcls@debug\typeout
1464 \let\pkgcls@debug\@gobble
1465 </tracerollback>

```

(End definition for \pkgcls@debug.)

\requestedLaTeXdate The macro (!) \requestedLaTeXdate holds the globally requested rollback date (via `latexrelease`) or zero if no such request was made.

```

1466 \def\requestedLaTeXdate{0}

```

(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 `TEX` largest legal number. While loading a file this can be used to provide conditionals that select code based on the request.

```

1467 \ifx\pkgcls@targetdate\@undefined

```

```

1468 \newcount\pkgcls@targetdate
1469 \fi
1470 \let\pkgcls@targetlabel\empty
1471 \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.

```

1472 \let\pkgcls@candidate\empty
1473 \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.

```

1474 \ifx\load@onefilewithoptions\undefined
1475 \let\load@onefilewithoptions\onefilewithoptions
1476 \def\onefilewithoptions#1[#2] [#3]#4{%

```

First a bit of tracing normally disabled.

```

1477 {*tracerollback}
1478 \pkgcls@debug{--- File loaded request (\noexpand\usepackage or ...)}`}
1479 \pkgcls@debug{@spaces 1: #1}`}
1480 \pkgcls@debug{@spaces 2: #2}`}
1481 \pkgcls@debug{@spaces 3: #3}`}
1482 \pkgcls@debug{@spaces 4: #4}`}
1483
```

Two of the arguments are needed later on in error/warning messages so we save them.

```

1484 \def\pkgcls@name{#1}%
1485 \def\pkgcls@arg {#3}%

```

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.

```
1486 \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:

```
1487 \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.

```

1488 \begingroup
1489 \edef\reserved@a{%
1490 \endgroup

```

```

1491 \unexpanded{\load@onefilewithoptions#1[#2]%
1492 [\pkgcls@mindate]%
1493 \unexpanded{#4}%
1494 \reserved@a
1495 }
1496 \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>\globalrollbackdate-as-number</code> | <code>\empty</code>              | <code>\empty</code>          |
| <code>(date)</code>      | <code>\globalrollbackdate-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>\globalrollbackdate-as-number</code> | <code>\empty</code>              | <code>(other)</code>         |

where `\globalrollbackdate-as-number` is a date request given via `latexrelease` or if there wasn’t one 0.

```
1497 \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.

```

1498 \ifx\@nil#1\@nil
1499 \pkgcls@targetdate\requestedLaTeXdate\relax
1500 \let\pkgcls@targetlabel\@empty
1501 \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.

```

1502 \else
1503 \pkgcls@parse@date@arg@#1=\@nil\relax
1504 \fi
1505 }

```

The actual parsing work then happens in `\pkgcls@parse@date@arg@`:

```
1506 \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.

```
1507 \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.

```
1508 \ifx\@nil#1\@nil
1509 \@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 `=`).

```
1510 \ifnum \pkgcls@targetdate=\z@
1511 \pkgcls@targetdate\@ne
1512 \def\pkgcls@innerdate{\maxdimen}%
1513 \pkgcls@parse@date@arg@version#2%
1514 \else
1515 \edef\pkgcls@innerdate{\the\pkgcls@targetdate}%
1516 \fi
1517 \let\pkgcls@mindate\@empty
1518 \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`):

```
1519 \requestedLaTeXdate\relax
1520 \let\pkgcls@targetlabel\@empty
1521 \def\pkgcls@innerdate{\maxdimen}%
1522 \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.

```
1523 \ifnum \pkgcls@targetdate > \z@
1524 \ifnum \@parse@version0#1//00\@nil > \pkgcls@targetdate
1525 \@latex@warning@no@line{Suspicious rollback/min-date date given\MessageBreak
1526 A minimal date of #1 has been specified for
1527 \@cls@pkg\MessageBreak '\pkgcls@name'.\MessageBreak
1528 But this is in conflict
1529 with a rollback request to \requestedpatchdate}
1530 \fi
1531 \fi
1532 \fi
1533 }
```

Strip off the trailing `=` and assign the version name to `\pkgcls@targetlabel`.

```
1534 \def\pkgcls@parse@date@arg@version#1=%
1535 \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  $\langle\text{pkg/cls-name}\rangle-\langle\text{date}\rangle.\langle\text{extension}\rangle$  but this is not enforced and through this argument one can overwrite it.

```
1536 \def\DeclareRelease#1#2#3{%
1537 \ifnum\pkgcls@targetdate>\z@ % some sort of rollback request
1538 (*tracerollback)
1539 \pkgcls@debug{---\string\DeclareRelease:{}}
1540 \pkgcls@debug{\@spaces 1: #1}%
1541 \pkgcls@debug{\@spaces 2: #2}%
1542 \pkgcls@debug{\@spaces 3: #3}%
1543
```

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.

```
1544 \ifx\@nil#2\@nil
1545 \ifnum\pkgcls@targetdate=\@ne % named request
1546 \def\reserved@a{#1}%
1547 \ifx\pkgcls@targetlabel\reserved@a
1548 \pkgcls@use@this@release{#3}{}%
1549 (*tracerollback)
1550 \else
1551 \pkgcls@debug{Label doesn't match}%
1552
```

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.

```
1560 \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.

```
1561 \ifnum\@parse@version#2//00\@nil
1562 >\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 exists. So we make a complained to the user.

```
1563 \ifx\pkgcls@candidate\@empty
1564 \pkgcls@rollbackdate@error{#2}%
1565 \pkgcls@use@this@release{#3}{#2}%
1566
```

```

1567 \pkgcls@use@this@release\pkgcls@candidate
1568 \pkgcls@releasedate
1569 \fi
1570 \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.

```

1571 \def\pkgcls@candidate{\#3}%
1572 \def\pkgcls@releasedate{\#2}%
1573 {*tracerollback}
1574 \pkgcls@debug{New candidate: #3}%
1575 {/tracerollback}
1576 \fi
1577 \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.

```

1578 \def\reserved@a{\#1}%
1579 \ifx\pkgcls@targetlabel\reserved@a
1580 \pkgcls@use@this@release{\#3}{\#2}%
1581 {*tracerollback}
1582 \else
1583 \pkgcls@debug{Label doesn't match}%
1584 {/tracerollback}
1585 \fi
1586 \fi
1587 \fi
1588 \fi
1589 }
```

*(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.

```
1590 \def\pkgcls@use@this@release{\#1}{}
```

Before that we record the selection made inside the transcript.

```
1591 \pkgcls@show@selection{\#1}{}
```

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`!

```

1592 \pkgcls@targetdate\z@
1593 \@@input #1\relax
1594 \endinput
1595 }
```

---

<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@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.

```
1596 \def\pkgcls@show@selection#1#2{%
1597 {*tracer rollback}
1598 \pkgcls@debug{Result: use #1}%
1599 //tracer rollback
1600 \GenericInfo
1601 {\@spaces\@spaces\space}{Rollback for
1602 \@cls@pkg\space'\@currname' requested ->
1603 \ifnum\pkgcls@targetdate>\@ne
1604 date
1605 \ifnum\requestedLaTeXdate=\pkgcls@targetdate
1606 \requestedpatchdate
1607 \else
1608 \expandafter\@gobble\pkgcls@arg
1609 \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.

```
1610 Best approximation is
1611 \else
1612 version '\pkgcls@targetlabel'.\MessageBreak
1613 This corresponds to
1614 \fi
1615 \ifx\@nil#2\@nil
1616 a special release%
1617 \else
1618 the release introduced on #%
1619 \fi
1620 \@gobble}%
1621 }
```

(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.

```
1622 \def\pkgcls@rollbackdate@error#1{%
1623 \@latex@error{Suspicious rollback date given}%
1624 {The \@cls@pkg\space'\@currname' has no rollback data
1625 before #1 which\MessageBreak
1626 is after your requested rollback date --- so
1627 something may be wrong here.\MessageBreak
1628 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.

```
1629 \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).

```

1630 \ifnum\pkgcls@targetdate>\z0 % some sort of rollback request
1631 (*tracerrollback)
1632 \pkgcls@debug{---DeclareCurrentRelease}%
1633 \pkgcls@debug{ 1: #1}%
1634 \pkgcls@debug{ 2: #2}%
1635 (/tracerrollback)

```

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`.

```

1636 \ifnum\pkgcls@targetdate>\@ne % a date request
1637 \ifnum@\parse@version#2//00@nil
1638 >\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).

```

1639 \ifx\pkgcls@candidate@\empty
1640 \pkgcls@rollbackdate@error{#2}%
1641 \else
1642 \pkgcls@use@this@release\pkgcls@candidate
1643 \pkgcls@releasedate
1644 \fi

```

Otherwise the current file is the right release, so we record that in the transcript and then carry on.

```

1645 \else
1646 \pkgcls@show@selection{current version}{#2}%
1647 \fi
1648 \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.

```

1649 \def\reserved@a{#1}%
1650 \ifx\pkgcls@targetlabel\reserved@a
1651 \pkgcls@show@selection{current version}{#2}%
1652 \else
1653 \@latex@error{Requested version '\pkgcls@targetlabel' for
1654 '@cls@pkg@space'@\currname' is unknown}\@ehc
1655 \fi
1656 \fi
1657 \fi
1658 }

```

*(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.

```
1659 \DeclareRobustCommand\IfTargetDateBefore[1]{%
1660 \ifnum\pkc@innerdate <%
1661 \expandafter\@parse@version\expandafter0#1//00\@nil
1662 \typeout{Exclude code introduced on #1}%
1663 \expandafter\@firstoftwo
1664 \else
1665 \typeout{Include code introduced on #1}%
1666 \expandafter\@secondoftwo
1667 \fi
1668 }

(End definition for \IfTargetDateBefore.)
```

```
1669
```

## 6 After Preamble

Finally we declare a package that allows all the commands declared above to be `\onlypreamble` to be used after `\begin{document}`.

```
1670 <!*afterpreamble>
1671 \NeedsTeXFormat{LaTeX2e}
1672 \ProvidesPackage{pkgindoc}
1673 [2020-08-08 v1.3m Package Interface in Document (DPC)]
1674 \def\reserved@a{\do\@classoptionslist\do\@filec\@ntents\relax}%
1675 \gdef\@preamblecmds{\@empty}%
1676 \expandafter\reserved@a\@preamblecmds\relax
1677
```

# File T

## 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.

---

```
package/before
package/after
package/.../before
package/.../after
class/before
class/after
class/.../before
class/.../after
```

---

These are:

**package/before, package/after** These hooks are called for each package being loaded.

**package/⟨name⟩/before, package/⟨name⟩/after** These hooks are additionally called if the package name is ⟨name⟩ (without extension).

**class/before, class/after** These hooks are called for each class being loaded.

**class/⟨name⟩/before, class/⟨name⟩/after** These hooks are additionally called if the class name is ⟨name⟩ (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/variorref/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 {\langle file\rangle} {\langle replacement-file\rangle}
\undeclare@file@substitution \undeclare@file@substitution {\langle file\rangle}
```

---

If  $\langle file \rangle$  is requested for loading replace it with  $\langle replacement-file \rangle$ .  $\text{\CurrentFile}$  remains pointing to  $\langle file \rangle$  but  $\text{\CurrentFileUsed}$  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  $\text{\undeclare@file@substitution}$  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 {\langle package\rangle} {\langle alternate-code\rangle}
\reenable@package@load \reenable@package@load {\langle package\rangle}
```

---

If  $\langle package \rangle$  is requested do not load it but instead run  $\langle alternate-code \rangle$  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 836.)

## 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@tfilef@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>

```

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 839.)

```
265 <@@=〉
266 \ExplSyntaxOff
```

## 2.6 Selecting a file (`\set@curr@file`)

`\set@curr@file` Now we hook into `\set@curr@file` to resolve a possible file substitution, and add `\Expl@@@filehook@set@curr@file@onNN` at the end, after `\curr@file` is set.

`\curr@file@reqd` 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{2021/06/01}%
270 <latexrelease> {\set@curr@file}{Setting current file name}%
271 \def\set@curr@file#1{%
272 \begingroup
273 \escapechar\m@ne
274 \let\protect\string
275 \edef~{\string~}%
276 \expandafter\def\csname\expandafter\endcsname
277 \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@CTF` 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.

```
278 \Expl@@@filehook@if@no@extension@N\TF{#1}%
279 { \tempswat{true}{\tempswafalse} }%
280 \kernel@make@file@csname\curr@file
281 \Expl@@@filehook@resolve@file@subst@w {#1}%
282 \Expl@@@filehook@if@file@replaced@CTF
283 { \kernel@make@file@csname\curr@file@reqd
284 \Expl@@@filehook@normalize@file@name@w{#1}%
285 \if@tempswa \Expl@@@filehook@drop@extension@N\curr@file@reqd \fi }%
286 { \if@tempswa \Expl@@@filehook@drop@extension@N\curr@file \fi }
```

```

287 \global\let\@curr@file@reqd\@curr@file}%
288 \Expl@@@filehook@clear@replacement@flag@@
289 \endgroup}
290 {/2ekernel | latexrelease}
291 \end{IncludeInRelease}

292 \end{IncludeInRelease}[2020/10/01]%
293 \set@curr@file}{Setting current file name}%
294 \def\set@curr@file#1{%
295 \begingroup
296 \escapechar\m@ne
297 \expandafter\def\csname\expandafter\endcsname
298 \expandafter{\csname\endcsname}%
299 \Expl@@@filehook@if@no@extension@onTF{#1}%
300 {\@tempswatru{}}{\@tempswafalse}%
301 \kernel@make@file@csname@\curr@file
302 \Expl@@@filehook@resolve@file@subst@ow {#1}%
303 \Expl@@@filehook@if@file@replaced@OTF
304 {\@kernel@make@file@csname@\curr@file@reqd
305 \Expl@@@filehook@normalize@file@name@ow{#1}%
306 \if@tempswa \Expl@@@filehook@drop@extension@N@\curr@file@reqd \fi}%
307 {\if@tempswa \Expl@@@filehook@drop@extension@N@\curr@file \fi
308 \global\let\@curr@file@reqd\@curr@file}%
309 \Expl@@@filehook@clear@replacement@flag@@
310 \endgroup}
311 \end{IncludeInRelease}

312 \end{IncludeInRelease}[2019/10/01]%
313 \set@curr@file}{Setting current file name}%
314 \def\set@curr@file#1{%
315 \begingroup
316 \escapechar\m@ne
317 \xdef@\curr@file{%
318 \expandafter\expandafter\expandafter\unquote@name
319 \expandafter\expandafter\expandafter{%
320 \expandafter\string
321 \csname\@firstofone#1\empty\endcsname}}%
322 \endgroup
323 \end{IncludeInRelease}

324 \end{IncludeInRelease}[0000/00/00]%
325 \set@curr@file}{Setting current file name}%
326 \let\set@curr@file@\undefined
327 \end{IncludeInRelease}
328 {*2ekernel}

(End definition for \set@curr@file, \curr@file, and \curr@file@reqd.)

```

```
\filehook@set@CurrentFile
\kernel@make@file@csname
\set@curr@file@aux
```

*Todo: This should get internalized using @Expl@ names*

```

330 {/2ekernel}
331 {*2ekernel | latexrelease}
332 \end{IncludeInRelease}[2020/10/01]%
333 \set@curr@file}{Make file csname}%

```

```

334 \def\@kernel@make@file@csname#1#2#3{%
335 \xdef#1{\expandafter\@set@curr@file@aux
336 \csname\expandafter#2\@firstofone#3\@nil\endcsname}}

```

This auxiliary compares  $\langle\text{filename}\rangle$  with  $\text{\csname}\endcsname$  to check if the empty  $.tex$  file was requested.

```

337 \def\@set@curr@file@aux#1{%
338 \expandafter\ifx\csname\endcsname#1%
339 .tex\else\string#1\fi}

```

Then we call  $\text{@expl@@@filehook@set@curr@file@nNN}$  once for  $\text{@curr@file}$  to set  $\text{\CurrentFile(Path)Used}$  and once for  $\text{@curr@file@reqd}$  to set  $\text{\CurrentFile(Path)}$ . Here too the slower route is only used if a substitution happened, but here  $\text{@expl@@@filehook@if@file@replaced@@TF}$  can't be used because the flag is reset at the  $\text{\endgroup}$  above, so we check if  $\text{@curr@file}$  and  $\text{@curr@file@reqd}$  differ. This macro is issued separate from  $\text{\set@curr@file}$  because it changes  $\text{\CurrentFile}$ , and side-effects would quickly get out of control.

```

340 \def\@filehook@set@CurrentFile{%
341 \text{@expl@@@filehook@set@curr@file@nNN}{\text{@curr@file}}%
342 \text{\CurrentFileUsed}\text{\CurrentFilePathUsed}
343 \ifx\text{@curr@file@reqd}\text{@curr@file}
344 \text{\let}\text{\CurrentFile}\text{\CurrentFileUsed}
345 \text{\let}\text{\CurrentFilePath}\text{\CurrentFilePathUsed}
346 \else
347 \text{@expl@@@filehook@set@curr@file@nNN}{\text{@curr@file@reqd}}%
348 \text{\CurrentFile}\text{\CurrentFilePath}
349 \fi}
350 {/2ekernel | latexrelease}
351 (latexrelease)\text{EndIncludeInRelease}
352 {*2ekernel}

```

(End definition for  $\text{@filehook@set@CurrentFile}$ ,  $\text{@kernel@make@file@csname}$ , and  $\text{@set@curr@file@aux}$ .)

$\text{@@_set_curr_file:nNN}$  When inputting a file,  $\text{\set@curr@file}$  does a file lookup (in  $\text{\input@path}$  and  $\text{\l_file_search_path_seq}$ ) and returns the actual file name ( $\langle\text{base}\rangle$  plus  $\langle\text{ext}\rangle$ ) in  $\text{\CurrentFileUsed}$ , and in case there's a file substitution, the requested file in  $\text{\CurrentFile}$  (otherwise both are the same). Only the base and extension are returned, regardless of the input (both  $\text{path/to/file.tex}$  and  $\text{file.tex}$  end up as  $\text{file.tex}$  in  $\text{\CurrentFile}$ ). The path is returned in  $\text{\CurrentFilePath}$ , in case it's needed.

```

353 {/2ekernel}
354 {*2ekernel | latexrelease}
355 (latexrelease)\text{IncludeInRelease}[2020/10/01]%
356 (latexrelease) {\text{@@_set_curr_file:nNN}{\text{Set curr file}}%}
357 \ExplSyntaxOn
358 \text{@=filehook}
359 \text{cs_new_protected:Npn }\text{__filehook_set_curr_file:nNN} #1
360 {
361 \text{exp_args:Nf }\text{__filehook_file_parse_full_name:nN} {#1}
362 \text{__filehook_set_curr_file_assign:nnnNN}
363 }
364 \text{cs_new_protected:Npn }\text{__filehook_set_curr_file_assign:nnnNN} #1 #2 #3 #4 #5
365 {
366 \text{str_set:Nn} #5 {#1}

```

```

367 \str_set:Nn #4 {#2#3}
368 }
369 \ExplSyntaxOff
370 {/2ekernel | latexrelease}
371 \EndIncludeInRelease
372 (*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>}`.

```

373 {/2ekernel}
374 (*2ekernel | latexrelease)
375 \EndIncludeInRelease{2020/10/01}%
376 \EndIncludeInRelease{Replace files detect loops}%
377 \ExplSyntaxOn
378 \cs_new:Npn __filehook_resolve_file_subst:w #1 \@nil
379 { __filehook_file_parse_full_name:nN {#1} __filehook_file_subst_begin:nnn }
380 \cs_new:Npn __filehook_normalize_file_name:w #1 \@nil
381 { __filehook_file_parse_full_name:nN {#1} __filehook_file_name_compose:nnn }
382 \cs_new:Npn __filehook_file_name_compose:nnn #1 #2 #3
383 { \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.

```

384 \flag_new:n { __filehook_file_replaced }
385 \cs_new:Npn __filehook_if_file_replaced:TF #1 #2
386 { \flag_if_raised:nTF { __filehook_file_replaced } {#1} {#2} }
387 \cs_new_protected:Npn __filehook_clear_replacement_flag:
388 { \flag_clear:n { __filehook_file_replaced } }

```

First off, start by checking if the current file (`<name>+<ext>`) has a declared substitution. If not, then just put that as the name (including a possible `<path>` 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.

```

389 \cs_new:Npn __filehook_file_subst_begin:nnn #1 #2 #3
390 {
391 __filehook_file_subst_tortoise_hare:nn {#2#3} {#2#3}
392 { __filehook_file_name_compose:nnn {#1} {#2} {#3} }
393 }
394 \ExplSyntaxOff
395 {/2ekernel | latexrelease}
396 \EndIncludeInRelease
397 (*2ekernel)

```

### 2.7.1 The Tortoise and Hare algorithm

If there is a substitution (*true*) 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@<file>` and `\@file-subst@<file-subst@<file>>`; that is, the substitution of `<file>` 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.

```

398 </2ekernel>
399 {*2ekernel | latexrelease}
400 <latexrelease>\IncludeInRelease{2020/10/01}%
401 <latexrelease> {_filehook_file_subst_tortoise_hare:nn}{Tortoise and Hare}%
402 \ExplSyntaxOn
403 \cs_new:Npn _filehook_file_subst_tortoise_hare:nn #1 #2 #3
404 {
405 \cs_if_exist:cTF { @file-subst@ #2 }
406 {
407 \flag_if_raised:nF { _filehook_file_replaced }
408 { \flag_raise:n { _filehook_file_replaced } }
409 \cs_if_exist:cTF { @file-subst@ \use:c { @file-subst@ #2 } }
410 {
411 _filehook_file_subst_loop:cc
412 { @file-subst@ #1 }
413 { @file-subst@ \use:c { @file-subst@ #2 } }
414 }
415 { \use:c { @file-subst@ #2 } }
416 }
417 { #3 }
418 }
419 \cs_new:Npn _filehook_file_subst_loop:NN #1 #

```

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.

```
419 \cs_new:Npn _filehook_file_subst_loop:NN #1 #2
```

```

420 {
421 \token_if_eq_meaning:NNTF #1 #2
422 {
423 .tex
424 __filehook_file_subst_cycle_error:cN { @file-subst@ #1 } #1
425 }
426 { __filehook_file_subst_tortoise_hare:nn {#1} {#2} {#2} }
427 }
428 \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.

```

429 \cs_new:Npn __filehook_file_subst_cycle_error:NN #1 #2
430 {
431 \msg_expandable_error:nnff { latex2e } { file-cycle }
432 {#1} { \use:c { @file-subst@ #1 } }
433 \token_if_eq_meaning:NNF #1 #2
434 { __filehook_file_subst_cycle_error:cN { @file-subst@ #1 } #2 }
435 }
436 \cs_generate_variant:Nn __filehook_file_subst_cycle_error:NN { c }

And the error message:

437 \msg_new:nnn { latex2e } { file-cycle }
438 { File-loop!~#1~replaced~by~#2... }

```

(End definition for `\__filehook_resolve_file_subst:w` and others.)

```

439 \ExplSyntaxOff
440 ⟨/2ekernel | latexrelease⟩
441 ⟨latexrelease⟩\EndIncludeInRelease
442 ⟨*2ekernel⟩
443 ⟨@@=⟩

```

## 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 \disable@package@load defines \@pkg-disable@⟨package⟩ to expand to some code #2
\reenable@package@load instead of loading the package.

\@disable@packageload@do
444 ⟨/2ekernel⟩
445 ⟨*2ekernel | latexrelease⟩
446 ⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
447 ⟨latexrelease⟩ {\@disable@package@load}{Disable packages}%
448 \def\@disable@package@load#1#2{%
449 \global\@namedef{@pkg-disable@#1.\@pkextension}{#2}}
450 \def\@disable@packageload@do#1#2{%
451 \@ifundefined{@pkg-disable@#1}{#2}{%
452 {\@nameuse{@pkg-disable@#1}}}

```

\reenable@package@load undefines \@pkg-disable@*package* to reallow loading a package.

```

453 \def\reenable@package@load#1{%
454 \global\expandafter\let
455 \csname @pkg-disable@\#1.\@pkgextension \endcsname \undefined}
456 </2ekernel | latexrelease>
457 <latexrelease>\EndIncludeInRelease
458 <latexrelease>\IncludeInRelease{0000/00/00}%
459 <latexrelease> {\@disable@package@load}{Disable packages}%
460 <latexrelease>
461 <latexrelease>\let\@disable@package@load \undefined
462 <latexrelease>\let\@disable@packageload@do \undefined
463 <latexrelease>\let\reenable@package@load \undefined
464 <latexrelease>\EndIncludeInRelease
465 (*2ekernel)

```

(*End definition for \disable@package@load, \reenable@package@load, and \@disable@packageload@do. These functions are documented on page 839.*)

## 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>E</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>E</sub> syntax. We have to unset the @@ since we want double “at” sign in place of double underscores.

```

466 <@@=>
467 </2ekernel>
468 <*2ekernel | latexrelease>
469 <latexrelease>\IncludeInRelease{2020/10/01}%
470 <latexrelease> {\@expl@@@filehook@if@no@extension@nTF}{2e tmp interfaces}%
471 \ExplSyntaxOn
472 \cs_new_eq:NN \@expl@@@filehook@if@no@extension@nTF
473 __filehook_if_no_extension:nTF
474 \cs_new_eq:NN \@expl@@@filehook@set@curr@file@nNN
475 __filehook_set_curr_file:nNN
476 \cs_new_eq:NN \@expl@@@filehook@resolve@file@subst@w
477 __filehook_resolve_file_subst:w
478 \cs_new_eq:NN \@expl@@@filehook@normalize@file@name@w
479 __filehook_normalize_file_name:w
480 \cs_new_eq:NN \@expl@@@filehook@if@file@replaced@TF
481 __filehook_if_file_replaced:TF
482 \cs_new_eq:NN \@expl@@@filehook@clear@replacement@flag@%
483 __filehook_clear_replacement_flag:

```

```

484 \cs_new_eq:NN \Expl@@@filehook@drop@extension@@N
485 __filehook_drop_extension:N
486 \cs_new_eq:NN \Expl@@@filehook@file@push@@
487 __filehook_file_push:
488 \cs_new_eq:NN \Expl@@@filehook@file@pop@@
489 __filehook_file_pop:
490 \cs_new_eq:NN \Expl@@@filehook@file@pop@assign@@nnnn
491 __filehook_file_pop_assign:nnnn
492 \ExplSyntaxOff

```

This one specifically has to be undefined because it is left over in the input stream from `\InputIfFileExists` and executed when `\textrerelease` 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 `\textrerelease`.

```

493 </2ekernel | \textrerelease>
494 <\textrerelease>\EndIncludeInRelease
495 <\textrerelease>
496 <\textrerelease>\IncludeInRelease{0000/00/00}%
497 <\textrerelease> {\Expl@@@filehook@if@no@extension@nTF}{2e tmp interfaces}%
498 <\textrerelease>\let\Expl@@@filehook@file@pop@@\relax
499 <\textrerelease>\EndIncludeInRelease
500 <*2ekernel>

```

This ends the kernel code in this file.

```
501 </2ekernel>
```

### 3 A sample package for structuring the log output

```

502 <*structuredlog>
503 <@=filehook>
504 \ProvidesExplPackage
505 {structuredlog}{\ltfilehookdate}{\ltfilehookversion}
506 {Structuring the TeX transcript file}

```

`\g_filehook_nesting_level_int` Stores the current package nesting level.

```
507 \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.

```
508 \int_gset:Nn \g_filehook_nesting_level_int
509 { (\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.

```

510 \cs_new_protected:Npn _filehook_log_file_record:n #1
511 {
512 \str_if_eq:nnT {#1} {START} { \int_gincr:N \g_filehook_nesting_level_int }
513 \iow_term:x
514 {

```

```

515 \prg_replicate:nn { \g__filehook_nesting_level_int } { = } ~
516 (LEVEL ~ \int_use:N \g__filehook_nesting_level_int \c_space_tl #1) ~
517 \CurrentFileUsed

```

If there was a file replacement, show that as well:

```

518 \str_if_eq:NNF \CurrentFileUsed \CurrentFile
519 { ~ (\CurrentFile \c_space_tl requested) }
520 \iow_newline:
521 }
522 \str_if_eq:nnT {#1} {STOP} { \int_gdecr:N \g__filehook_nesting_level_int }
523 }

```

Now just hook the macro above in the generic `file/before...`

```
524 \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.

```

525 \AddToHookNext{file/after}
526 { \AddToHook{file/after}{ __filehook_log_file_record:n { STOP } } }

```

(End definition for `\__filehook_log_file_record:n`.)

```

527 <@C=›
528 </structuredlog>

```

## 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:

```

529 <*2ekernel›
530 \declare@file@substitution{atveryend.sty}{atveryend-ltx.sty}
531 </2ekernel›

```

Here is the package file we point to:

```

532 <*atveryend-ltx›
533 \ProvidesPackage{atveryend-ltx}
534 [2020/08/19 v1.0a
535 Emulation of the original atveryend package^^Jwith kernel methods]

```

Here are new definitions for its interfaces now pointing to the hooks in `\enddocument`

```
536 \newcommand\AfterLastShipout {\AddToHook{\enddocument}{afterlastpage}}
```

```
537 \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`.

```
538 \newcommand\AtEndAfterFileList{\AddToHook{\enddocument}{info}}
```

```
539 \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.

```
540 \ExplSyntaxOn
541 \newcommand\BeforeClearDocument[1]
542 { \AtEndDocument{#1}
543 \atveryend@DEPRECATED{BeforeClearDocument \tl_to_str:n{#1}}
544 }
545 \cs_new:Npn\atveryend@DEPRECATED #1
546 {\iow_term:x{=====~DEPRECATED~USAGE~#1~=====}}
547 \ExplSyntaxOff
(End definition for \BeforeClearDocument.)
548 ⟨/atveryend-ltx⟩
```

# File U

## 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><</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`  
`\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.

As these two wrappers have been available for a long time we continue offering them. However, for new code we 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. For this reason we do not provide any other wrapper commands for the above hooks in the kernel.

### 1.4 Special commands for use inside the hooks

---

`\DiscardShipoutBox`  
`\shipout_discard:`

---

`\AddToHookNext {shipout/before} {...\DiscardShipoutBox...}`

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` `\g_shipout_READONLY_int`

```
\ifnum_READONLYSHIPOUTCOUNTER=...
\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` `\g_shipout_TOTALPAGES_int`

```
\arabic{TOTALPAGES}
\int_use:N \g_shipout_TOTALPAGES_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  
\AtBeginShipoutNext**

`\AtBeginShipout{\<code>}` ≡ `\AddToHook{shipout/before}{\<code>}`  
`\AtBeginShipoutNext{\<code>}` ≡ `\AddToHookNext{shipout/before}{\<code>}`

This is equivalent to filling the `shipout/before` hook by either using `\AddToHook` or `\AddToHookNext`, respectively.

---

**\AtBeginShipoutFirst  
\AtBeginShipoutDiscard**

The `atbegshi` names for `\AtBeginDvi` and `\DiscardShipoutBox`.

## 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>}` ≡ `\AddToHook{shipout/before}{\<code>}`

---

**\AtNextShipout**

`\AtNextShipout{\<code>}` ≡ `\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{\langle code\rangle} \AddToHook{shipout/background}{\put(1in,-1in){\langle code\rangle}}
```

`\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.

```
\AddThispageHook \AddThispageHook{\langle code\rangle} \AddToHookNext{shipout/background}{\put(1in,-1in){\langle code\rangle}}
```

The `\AddThispageHook` wrapper is similar but uses `\AddToHookNext`.

## 3 The Implementation

```
1 <@=shipout>
```

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 863.)

`\ShipoutBox` The box filled with the page to be shipped out (both L<sup>3</sup> and L<sup>A</sup>T<sub>E</sub>X 2<sub>&</sub> name).

```
23 \box_new:N \l_shipout_box
24 \cs_set_eq:NN \ShipoutBox \l_shipout_box
```

(End definition for `\ShipoutBox` and `\l_shipout_box`. These functions are documented on page 859.)

`\l__shipout_raw_box` The `\RawShipout` gets its own box but it is internal as there is no hook manipulation for it.

```
25 \box_new:N \l__shipout_raw_box
```

(End definition for `\l__shipout_raw_box`.)

`\__shipout_finalize_box`: For LuaT<sub>E</sub>X invoke the `pre_shipout_filter` callback.

```
26 \sys_if_engine_luatex:TF
27 {
28 \newluafunction __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 __shipout_finalize_box:] = function()
35 local~result = call('pre_shipout_filter', getbox(the \l_shipout_box))
36 if-not (result == true) then~
37 setbox(the \l_shipout_box, result~or~nil)
38 end~
39 end
40 }
41 }
42 \protected\luadef __shipout_finalize_box: \the __shipout_finalize_box:
43 } {
44 \cs_set_eq:NN __shipout_finalize_box: \scan_stop:
45 }
```

(End definition for `\__shipout_finalize_box`.)

`\__shipout_execute`: This is going to be the code run by `\shipout`. The code follows closely the ideas from atbegshi, so not documenting that here for now.

```
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 }
```

(End definition for `\__shipout_execute`.)

`\shipout` Overloading the `\shipout` primitive:

```
52 \cs_gset_eq:NN \shipout __shipout_execute:
```

(End definition for `\shipout`. This function is documented on page 858.)

\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
 \l__shipout_box
 62 { \hook_use:n {shipout/before} }
 63 { \hook_if_empty:nF {shipout/foreground}
 { __shipout_add_foreground_picture:
 { \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:
 { \@kernel@before@shipout@background
 \hook_use:n {shipout/background}
 \@kernel@after@shipout@background } }
 70 }
 71 }
 72 { \hook_use:n {shipout/foreground} }
 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 858.)

`\__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 859.)

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:` and `\__shipout_drop_firstpage_specials:` The `\__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 859.)

`\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 }
```

```

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 861.)

## 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 862.)

`\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 862.)

`\@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 [862](#).)

## 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 [861](#).)

**\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 [861](#).)

**\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 [863](#).)

## 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 to 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 {\AddToHook{shipout/lastpage}}
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 861.)

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 863.)

**\AtBeginShipoutInit**

Compatibility only, we aren't delaying ...

```
484 \let \AtBeginShipoutInit \empty
```

(End definition for \AtBeginShipoutInit. This function is documented on page 864.)

**\AtBeginShipout**

Filling hooks

```

485 \protected \def \AtBeginShipout {\AddToHook{shipout/before}}
486 \protected \def \AtBeginShipoutNext {\AddToHookNext{shipout/before}}

```

(End definition for \AtBeginShipout and \AtBeginShipoutNext. These functions are documented on page 864.)

**\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 864.)

**\AtBeginShipoutDiscard**

Just a different name.

```
489 \let \AtBeginShipoutDiscard \DiscardShipoutBox
```

(End definition for \AtBeginShipoutDiscard. This function is documented on page 864.)

**\AtBeginShipoutAddToBox**

We don't expose them.

```
490 \let \AtBeginShipoutAddToBox
491 \Expl@@@shipout@add@background@box@Nn
```

**\AtBeginShipoutUpperLeft**

```
492 \let \AtBeginShipoutAddToBoxForeground
493 \Expl@@@shipout@add@foreground@box@Nn
```

**\AtBeginShipoutUpperLeftForeground**

```

494 \let \AtBeginShipoutUpperLeft
495 \Expl@@@shipout@add@background@picture@Nn
496 \let \AtBeginShipoutUpperLeftForeground
497 \Expl@@@shipout@add@foreground@picture@Nn

```

(End definition for `\AtBeginShipoutAddToBox` and others. These functions are documented on page 863.)

### `\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 864.)

### `\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 V

## 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</code> ( <code>\penalty -10000 \vbox{}</code> ) <code>\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 861.)

**\@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@firstcolumn** 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 \c@mparbottom\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 =\m@ne
128 \ifdim \pagetotal <\topskip
129 \hbox{}%
130 \fi
131 \fi
132 \fi
133 \newpage
134 \write\m@ne{}%
135 \vbox{}%
136 \penalty -\OMi
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.

```

\@topnewpage
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 % \if@firstcolumn
281 % \else
282 % \@emptycol
283 % \fi
284 % \fi
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 \setbox\@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@fcolmade \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 \dblfloatafterplacement

```

```

402 \@makefcolumn\@deferlist
403 \@whilesw\if@fcolmade \fi{\@outputpage
404 \@makefcolumn\@deferlist}%
405 \endgroup
406 \else
407 \vbox{}\clearpage
408 \fi
409 \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.

```

410 \ifx\@deferlist\empty \else\clearpage \fi
411 \else
412 \setbox\@cclv\vbox{\box\@cclv\vfil}%
413 \@makecol\@opcol
414 \clearpage
415 \fi
416 }%
417 </2ekernel | latexrelease>
418 <latexrelease>\EndIncludeInRelease
419 <latexrelease>\IncludeInRelease{0000/00/00}{\@doclearpage}%
420 <latexrelease> {float order in 2-column}%
421 <latexrelease>\def \@doclearpage {%
422 <latexrelease> \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.

```

423 <latexrelease> \ifvbox\@kludgeins
424 <latexrelease> {\setbox \@tempboxa \box \@kludgeins}%
425 <*trace>
426 <latexrelease> \f@ttrace {kludgeins box made void}%
427 </trace>
428 <latexrelease> \fi
429 <latexrelease> \setbox\@tempboxa\vsplit\@cclv to\z@\unvbox\@tempboxa
430 <latexrelease> \setbox\@tempboxa\box\@cclv
431 <latexrelease> \xdef\@deferlist{\@toplist\@botlist\@deferlist}%
432 <latexrelease> \global \let \@toplist \empty
433 <latexrelease> \global \let \@botlist \empty
434 <latexrelease> \global \@colroom \colht
435 <latexrelease> \ifx \@currlist\empty
436 <latexrelease> \else
437 <latexrelease> \@latexerr{Float(s) lost}\@ehb
438 <latexrelease> \global \let \@currlist \empty
439 <latexrelease> \fi
440 <latexrelease> \@makefcolumn\@deferlist
441 <latexrelease> \@whilesw\if@fcolmade \fi
442 <latexrelease> {\@opcol\@makefcolumn\@deferlist}%
443 <latexrelease> \if@twocolumn
444 <latexrelease> \if@firstcolumn
445 <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 </trace>

```

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 \@mparbottom \z@ \global \@textfloatsheight \z@
477 \@floatplacement
478 }
479 </2ekernel | fltrace>

```

(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 \@shipoutsetup.

```
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 \@shipoutsetup.

```
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}}
\@combinedblfloats 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 \xtryfc \@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} \obitor \@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 \f1@trace{***Start addtobot}%
979 {/trace}
980 \getfpsbit 4\relax
981 {*trace}
982 \f1@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 \@tempswafalse
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 \f@l@trace{colroom (after-bot) = \the \@colroom}%
995 \f@l@trace{colnum (after-bot) = \the \@colnum}%
996 \f@l@trace{botnum (after-bot) = \the \@botnum}%
997 \f@l@trace{***Success: bot}%
998
```

 $\langle /trace \rangle$ 

```

999 \@inserttrue
1000 \fi
1001 {*trace}
1002 \else
1003 \f@l@trace{Fail: botnum = \the \@botnum:
1004 fpstype \the \@fpstype=ORD?}%
1005 \ifnum \@fpstype<\sixt@n
1006 \f@l@trace{ERROR: !b float not successful (addtobot)}%
1007 \fi
1008
```

 $\langle /trace \rangle$ 

```

1009 \fi
1010 \fi
1011 }

```

(End definition for \@addtobot.)

\@addtotoporbot Lots of changes.

```

1012 \def \@addtotoporbot {%
1013 {*trace}
1014 \f@l@trace{***Start addtotoporbot}%
1015
```

 $\langle /trace \rangle$ 

```

1016 \@getfpsbit \tw@
1017 {*trace}
1018 \f@l@trace{fpstype \ifodd \@tempcnta OK \else not \fi top:
1019 \the \@fpstype}%
1020
```

 $\langle /trace \rangle$ 

```

1021 \ifodd \@tempcnta
1022 \@flsetnum \@topnum
1023 \ifnum \@topnum>\z@
1024 \@tempswafalse
1025 \@flcheckspace \@toproom \@topl
1026 \if@tempswa
1027 \@bitor\@currtype{\@midlist\@botlist}%
1028 {*trace}
1029 \f@l@trace{(mid+bot)list: \@midlist, \@botlist:
1030 (addtotoporbot-before)}%
1031
```

 $\langle /trace \rangle$

```

1032 \if@test
1033 <*trace>
1034 \fl@trace{type already on list: mid or bot---sent to addtobot}%
1035 </trace>
1036 \else
1037 \@flupdates \@topnum \@toproom \@topl
1038 <*trace>
1039 \fl@trace{colroom (after-top) = \the \@colroom}%
1040 \fl@trace{colnum (after-top) = \the \@colnum}%
1041 \fl@trace{topnum (after-top) = \the \@topnum}%
1042 \fl@trace{***Success: top}%
1043 </trace>
1044 \@inserttrue
1045 \fi
1046 \fi
1047 <*trace>
1048 \else
1049 \fl@trace{Fail: topnum = \the \@topnum: fpstype
1050 \the \@fpstype=ORD?}%
1051 \ifnum \@fpstype<\sixt@n
1052 \fl@trace{ERROR: !t float not successful (addtotoporbot)}%
1053 \fi
1054 </trace>
1055 \fi
1056 \fi
1057 \@ifinsert
1058 \else
1059 <*trace>
1060 \fl@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 \fl@trace{***Start addtocurcol}%
1072 </trace>
1073 \@insertfalse
1074 \@setfloattypecounts
1075 \ifnum \@fpstype=8
1076 <*trace>
1077 \fl@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
```

```
1097
```

```
\reqcolroom \textmin
```

```
<*trace>
```

```
\fl@trace{ORD? textmin being used}%

```

```
</trace>
```

```
\fi
```

```
1102
```

```
\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
```

```
1109
```

```
\fisetnum \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
```

```
1121
```

```
\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 </trace>
1129 \addtobot
1130 \else
1131 <*trace>
1132 \fl@trace{fpstype \ifodd \tempcnta OK \else not \fi
1133 here: \the \fpstype}%
1134 </trace>
1135 \ifodd \count\currbox
1136 \advance \reqcolroom \intextsep
1137 \ifdim \colroom>\reqcolroom
1138 \global \advance \colnum \mone
1139 \global \advance \textfloatsheight \ht\currbox

```

This may sometimes give an overestimate.

```

1140 \global \advance \textfloatsheight 2\intextsep
1141 \cons \midlist \currbox
1142 <*trace>
1143 \fl@trace{***Success: here}%
1144 \fl@trace{textfloatsheight (after-here) =
1145 \the \textfloatsheight}%
1146 \fl@trace{colnum (after-here) = \the \colnum}%
1147 </trace>

```

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).

```

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 <*trace>

```

```

1163 \else
1164 \fl@trace{Fail---no room at 2nd test of colroom
1165 (addtocorcol \string\intextsep)}%
1166
```

```

1167 \fi
1168
```

```

1169 \if@insert
1170 \else

```

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
```

```

1173
```

```

1174
```

```

1175 \@addtotoporbot

```

```

1176
```

```

1177 <!2ekernel&!fltrace&!latexrelease>

```

```

1178
```

```

1179 \fl@trace{not here: sent to addtotoporbot}%

```

```

1180
```

```

1181 \@addtobot

```

```

1182
```

```

1183 \fl@trace{not here: sent to addtobot}%

```

```

1184 \fi

```

```

1185 \fi

```

```

1186
```

```

1187 \else

```

```

1188 \fl@trace{Fail: colnum = \the \colnum:

```

```

1189 fpstype \the \fpstype=ORD?}%

```

```

1190 \ifnum \fpstype<\sixt@n

```

```

1191 \fl@trace{ERROR: BANG float not successful (addtocurcol)}%

```

```

1192 \fi

```

```

1193
```

```

1194 \fi

```

```

1195
```

```

1196 \else

```

```

1197 \fl@trace{Fail---no room: fl box ht: \the \ht \currbox

```

```

1198 (addtocurcol)}%

```

```

1199
```

```

1200 \fi

```

```

1201 \fi

```

```

1202 \fi

```

```

1203 \if@insert

```

```

1204 \else

```

```

1205 \resetfps

```

```

1206
```

```

1207 \fl@trace{put on deferlist (addtocurcol)}%

```

```

1208
```

```

1209 \cons\deferlist\currbox

```

```

1210
```

```

1211 \fl@trace{deferlist: \deferlist: (addtocurcol-after)}%

```

```

1212
```

```

1213
```

```

1214
```

```

1215
```

```

1216
```

```

1217
```

```

1218
```

```

1219
```

```

1220
```

```

1221
```

```

1222
```

```

1223
```

```

1224
```

```

1225
```

```

1226
```

```

1227
```

```

1228
```

```

1229
```

```

1230
```

```

1231
```

```

1232
```

```

1233
```

```

1234
```

```

1235
```

```

1236
```

```

1237
```

```

1238
```

```

1239
```

```

1240
```

```

1241
```

```

1242
```

```

1243
```

```

1244
```

```

1245
```

```

1246
```

```

1247
```

```

1248
```

```

1249
```

```

1250
```

```

1251
```

```

1252
```

```

1253
```

```

1254
```

```

1255
```

```

1256
```

```

1257
```

```

1258
```

```

1259
```

```

1260
```

```

1261
```

```

1262
```

```

1263
```

```

1264
```

```

1265
```

```

1266
```

```

1267
```

```

1268
```

```

1269
```

```

1270
```

```

1271
```

```

1272
```

```

1273
```

```

1274
```

```

1275
```

```

1276
```

```

1277
```

```

1278
```

```

1279
```

```

1280
```

```

1281
```

```

1282
```

```

1283
```

```

1284
```

```

1285
```

```

1286
```

```

1287
```

```

1288
```

```

1289
```

```

1290
```

```

1291
```

```

1292
```

```

1293
```

```

1294
```

```

1295
```

```

1296
```

```

1297
```

```

1298
```

```

1299
```

```

1300
```

```

1301
```

```

1302
```

```

1303
```

```

1304
```

```

1305
```

```

1306
```

```

1307
```

```

1308
```

```

1309
```

```

1310
```

```

1311
```

```

1312
```

```

1313
```

```

1314
```

```

1315
```

```

1316
```

```

1317
```

```

1318
```

```

1319
```

```

1320
```

```

1321
```

```

1322
```

```

1323
```

```

1324
```

```

1325
```

```

1326
```

```

1327
```

```

1328
```

```

1329
```

```

1330
```

```

1331
```

```

1332
```

```

1333
```

```

1334
```

```

1335
```

```

1336
```

```

1337
```

```

1338
```

```

1339
```

```

1340
```

```

1341
```

```

1342
```

```

1343
```

```

1344
```

```

1345
```

```

1346
```

```

1347
```

```

1348
```

```

1349
```

```

1350
```

```

1351
```

```

1352
```

```

1353
```

```

1354
```

```

1355
```

```

1356
```

```

1357
```

```

1358
```

```

1359
```

```

1360
```

```

1361
```

```

1362
```

```

1363
```

```

1364
```

```

1365
```

```

1366
```

```

1367
```

```

1368
```

```

1369
```

```

1370
```

```

1371
```

```

1372
```

```

1373
```

```

1374
```

```

1375
```

```

1376
```

```

1377
```

```

1378
```

```

1379
```

```

1380
```

```

1381
```

```

1382
```

```

1383
```

```

1384
```

```

1385
```

```

1386
```

```

1387
```

```

1388
```

```

1389
```

```

1390
```

```

1391
```

```

1392
```

```

1393
```

```

1394
```

```

1395
```

```

1396
```

```

1397
```

```

1398
```

```

1399
```

```

1400
```

```

1401
```

```

1402
```

```

1403
```

```

1404
```

```

1405
```

```

1406
```

```

1407
```

```

1408
```

```

1409
```

```

1410
```

```

1411
```

```

1412
```

```

1413
```

```

1414
```

```

1415
```

```

1416
```

```

1417
```

```

1418
```

```

1419
```

```

1420
```

```

1421
```

```

1422
```

```

1423
```

```

1424
```

```

1425
```

```

1426
```

```

1427
```

```

1428
```

```

1429
```

```

1430
```

```

1431
```

```

1432
```

```

1433
```

```

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> \reqcolroom \textmin
1256 <latexrelease | fltrace | flafter> \fl@trace{ORD? textmin being used}%
1257 <latexrelease | fltrace | flafter> \fi
1258 <*trace>
1259 <latexrelease | fltrace | flafter> \advance \reqcolroom \ht\currbox
1260 <latexrelease | fltrace | flafter> \fl@trace{float size =
\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@
 \obitor\currtype\@deferlist
 \fl@trace{deferlist:
 \@deferlist: (addtocurcol-before)}%
 \if@test
 \fl@trace{type already on list:
 defer (addtocurcol)}%
 \else
 \obitor\currtype\@botlist
 \fl@trace{botlist: \@botlist:
 (addtocurcol-before)}%
 \if@test
 \fl@trace{type already on list:
 bot---sent to addtobot}%
 \else
 \addtobot
 \fl@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
 \fl@trace{***Success: here}%
 \fl@trace{textfloatsheight
 (after-here) =
 \the \@textfloatsheight}%
 \fl@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 \if@test
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 \fi
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 \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@l@trace{deferlist: \@deferlist:
1511 <| latexrelease |> fltrace) (addtonextcol-before)}%
1512 <| latexrelease |> fltrace)
1513 </trace>
1514 <| latexrelease |> fltrace) \if@test
1515 <| *trace>
1516 <| latexrelease |> fltrace) \f@l@trace{type already on list:
1517 <| latexrelease |> fltrace) defer (addtonextcol)}%
1518 </trace>
1519 <| latexrelease |> fltrace) \else
1520 <| *trace>
1521 <| latexrelease |> fltrace) \f@l@trace{sent to addtotoporbot
1522 <| latexrelease |> fltrace) (addtonextcol)}%
1523 </trace>
1524 <| latexrelease |> fltrace) \@addtotoporbot
1525 <| latexrelease |> fltrace) \fi
1526 <| latexrelease |> fltrace) \fi
1527 <| *trace>
1528 <| latexrelease |> fltrace) \else
1529 <| latexrelease |> fltrace) \f@l@trace{Fail---no room: f1 box ht:
1530 <| latexrelease |> fltrace) \the \ht \@currbox (addtonextcol)}%
1531 </trace>
1532 <| latexrelease |> fltrace) \fi
1533 <| latexrelease |> fltrace) \fi
1534 <| latexrelease |> fltrace) \fi
1535 <| latexrelease |> fltrace) \if@insert
1536 <| latexrelease |> fltrace) \else
1537 <| *trace>
1538 <| latexrelease |> fltrace) \f@l@trace{put back on deferlist
1539 <| latexrelease |> fltrace) (addtonextcol)}%
1540 </trace>
1541 <| latexrelease |> fltrace) \@cons\@deferlist\@currbox
1542 <| *trace>
1543 <| latexrelease |> fltrace) \f@l@trace{deferlist: \@deferlist:
1544 <| latexrelease |> fltrace) (addtonextcol-after)}%
1545 </trace>
1546 <| latexrelease |> fltrace) \fi
1547 <| *trace>
1548 <| latexrelease |> fltrace) \f@l@trace{End of addtonextcol --
1549 <| latexrelease |> fltrace) locally counts:}%
1550 <| latexrelease |> fltrace) \f@l@trace{col: \the \@colnum.
1551 <| latexrelease |> fltrace) top: \the \@topnum. bot: \the \@botnum.}%
1552 </trace>
1553 <| latexrelease |> fltrace) \endgroup
1554 <| *trace>
1555 <| latexrelease |> fltrace) \f@l@trace{End of addtonextcol --
1556 <| latexrelease |> fltrace) globally counts:}%
1557 <| latexrelease |> fltrace) \f@l@trace{col: \the \@colnum.
1558 <| latexrelease |> fltrace) top: \the \@topnum. bot: \the \@botnum.}%

```

```

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 \@insertfalse
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 \@flsetnum \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
 \tempswatru
\fi
\advance \@dbltoproom -\textmin
\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
\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 \caddmarginpar.)

### 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
```

```

1917 }

```

Here are two definitions from fltrace that make the above code work:

```

1917 \def \fl@traceval #1{\fl@trace{\string #1 = \the #1}}

```

```

1918 \def \fl@tracemessage #1{\let\@elt\empty\typeout{LaTeX2e: #1}}

```

```

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@tracef{\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 \f@trace{deferlist: \@deferlist}%
1986 \f@trace{dbltoplist: \@dbltoplist}%
1987 % next line only in old releases
1988 \f@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 `\f@trace` in that package.

```

2003 {*flafter}
2004 \providecommand\f@trace[1]{}
2005
```

*(End definition for `\f@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 \@setfloattypcounts.)*

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@on
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@on
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 \@tempsw@true
2165
```

`<*trace>`

```

2166 \else
2167 \fl@trace{Fail---no room (flcheckspace \string#1 \string#2)
2168 (fpstype \the \@fpstype=ORD?)}%
2169
```

`\@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 }
2181
```

`</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 }
2193
```

`</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\f@i---}%
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\f@i---}%
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
2230
```

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 }%
```

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.  
\@dblpsep  
\@dblfpbot  
2353 \newskip\@dblftop
2354 \newskip\@dblpsep
2355 \newskip\@dblfpbot
2356 \setlength\@dblftop{0\p@ \cplus 1fil}
2357 \setlength\@dblpsep{8\p@ \cplus 2fil}
2358 \setlength\@dblfpbot{0\p@ \cplus 1fil}

(End definition for \@dblftop, \@dblpsep, 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 W

### 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 X

## 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{EndIncludeInRelease}
128 \end{EndIncludeInRelease}

```

Then for the full rollback, just do nothing, since the function was already taken out of the rollback above.

```

129 \end{EndIncludeInRelease}{0000/00/00}{trace_stack_levels}%
130 \end{EndIncludeInRelease} {Lua trace_stack_levels function}%
131 \end{EndIncludeInRelease} Nothing here
132 \end{EndIncludeInRelease}
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\@addtolist##1{\xdef\@filelist{\@filelist,##1}}%
266 }
267 {\input{hyphen.ltx}}
268 \let\@addtolist\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\@inpenc@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`\^\^\I=10
505 \catcode`\^\^\J=12
506 \catcode`\^\^\L=13
507 \catcode`\^\^\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{"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.

```

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.

```

551 \DeclareRobustCommand{\MakeUppercase}[1]{%
552 \def\if{I}\def\j{J}%
553 \def\reserved@a##1##2{\let##1##2\reserved@a}%
554 \expandafter\reserved@a\@uclclist\reserved@b{\reserved@b\@gobble}%

```

Tell UTF-8 processing to process chars even though we are in an \protected@edef.

```

555 \let\UTF@two@octets@noexpand\empty
556 \let\UTF@three@octets@noexpand\empty
557 \let\UTF@four@octets@noexpand\empty
558 \protected@edef\reserved@a{\uppercase{#1}}%
559 \reserved@a
560 }

```

```

561 \DeclareRobustCommand{\MakeLowercase}[1]{{%
562 \def\reserved@a##1##2{\let##2##1\reserved@a}%
563 \expandafter\reserved@a\@uclclist\reserved@b{\reserved@b\@gobble}%
564 \let\UTF@two@octets@noexpand\empty
565 \let\UTF@three@octets@noexpand\empty
566 \let\UTF@four@octets@noexpand\empty
567 \protected@edef\reserved@a{\lowercase{#1}}%
568 \reserved@a
569 }
570 \def\@uclclist{\oe\OE\o\O\ae\AE
571 \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:

```

572 \protected@edef\MakeUppercase#1{\MakeUppercase{#1}}
573 \protected@edef\MakeLowercase#1{\MakeLowercase{#1}}
```

*(End definition for `\MakeUppercase`, `\MakeLowercase`, and `\@uclclist`.)*

## 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.

```

574 \%\\IfFileExists{ltpatch.ltx}
575 \% {\typeout{=====
576 \% Applying patch file ltpatch.ltx^^J%
577 \% ======}
578 \% \def\fmtversion@topatch{unknown}
579 \% \input{ltpatch.ltx}
580 \% \ifx\fmtversion\fmtversion@topatch
581 \% \ifx\patch@level@\undefined
582 \% \typeout{^^J^^J^^J%
583 \% !!!!!!!}
584 \% !! Patch file 'ltpatch.ltx' not suitable for this^^J%
585 \% !! version of LaTeX.^^J^^J%
586 \% !! Please check if initex found an old patch file:^^J%
587 \% !! --- if so, rename it or delete it, and redo the^^J%
588 \% !! initex run.^^J%
589 \% !!!!!!!}^^J}%

```

```

590 % \batchmode \@@end
591 %
592 % \def\fmtversion@topatch{0}%
593 % \ifx\fmtversion@topatch\patch@level\else
594 % \def\reserved@a{\typeout{\#1 patch level \patch@level}\#2\reserved@a{%
595 % \typeout{\#1 patch level \patch@level}\#2}
596 % \everyjob\expandafter\expandafter\expandafter{%
597 % \expandafter\reserved@a\the\everyjob\reserved@a}
598 % \let\reserved@a\relax
599 % \the\everyjob
600 % \fi
601 % \fi
602 % \else
603 % \typeout{^^J^^J^^J%
604 % !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^J%
605 % !! Patch file 'ltpatch.ltx' (for version <\fmtversion@topatch>)^J%
606 % !! is not suitable for version <\fmtversion> of LaTeX.^J%
607 % !! Please check if initex found an old patch file:^J%
608 % !! --- if so, rename it or delete it, and redo the^J%
609 % !! initex run.^J%
610 % !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^J}%
611 % \batchmode \@@end
612 % \fi
613 % \let\fmtversion@topatch\relax
614 % }{}}

```

## 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.*

```

615 \let\reserved@a\@filelist
616 \let\reserved@b=\@undefined
617 \let\reserved@c=\@undefined
618 \let\reserved@d=\@undefined
619 \let\reserved@e=\@undefined
620 \let\reserved@f=\@undefined

```

(End definition for \reserved@a and \reserved@b.)

\toks

```

621 \toks0{}
622 \toks2{}
623 \toks4{}
624 \toks6{}
625 \toks8{}

```

(End definition for \toks.)

\errhelp Empty the error help message, which may have some rubbish:  
 626 \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`.

```
627 \def\@providesfile#1[#2]{%
628 \wlog{File: #1 #2}%
629 \expandafter\xdef\csname ver@#1\endcsname{#2}%
630 }
```

(*End definition for \@providesfile.*)

- \@filelist \@addtofilelist Reset \@filelist so files input while making the format are not listed. The list built up 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.

```
631 \let\@filelist\gobble
632 \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).

```
633 \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.

```
634 \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.

```
635 \makeatother
636 \errorstopmode
637 \dump
638 </2ekernel>
```

# Change History

1985-11-04 ltmath.dtx LaTeX2.09		\mathversion: Test if version defined added. . . . .	412
General: produce warning message if line extends into margin. Doesn't warn about formula overprinting equation number. . . . .	638		
1989-04-10 ltfssbas.dtx v1.0a			
General: Starting with version numbers! \ifmmode added in \math@group . . . . .	401		
1989-04-10 ltfssbas.dtx v1.0b			
General: \preload@sizes added. . . . .	401		
\wrong@fontshape changed to define substitution font/shape macro. . . . .	401		
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. . . . .	509		
1989-04-14 ltfssbas.dtx v1.0c			
General: More documentation added. . . . .	401		
1989-04-15 ltfssini.dtx v1.0b			
General: \mathfontset renamed to \mathversion. . . . .	509		
1989-04-19 ltfssbas.dtx v1.0d			
General: Even more doc. . . . .	401		
1989-04-21 ltfssbas.dtx v1.0e			
General: Documentation is fun! Parameters of \define@mathalphabet changed. . . . .	401		
1989-04-21 ltfssini.dtx v1.0c			
General: Changed to conform to fam.tex. . . . .	509		
1989-04-23 ltfssbas.dtx v1.0f			
General: % in \getanddefinefonts added. . . . .	401		
1989-04-26 ltfssini.dtx v1.0d			
General: \xpt added. . . . .	509		
1989-04-27 ltfssbas.dtx v1.0g			
General: Documentation revised. . . . .	401		
1989-04-27 ltfssini.dtx v1.0e			
General: Definitions of L <sup>A</sup> T <sub>E</sub> X symbols corrected. . . . .	509		
1989-04-29 ltfssbas.dtx v1.0h			
General: Documented problem with \halign, and \noalign . . . . .	401		
		\mathversion: Test if version defined added. . . . .	412
1989-04-29 ltfssbas.dtx v1.0i			
General: Removed the \halign \noalign correction (wasn't bugfree) . . . . .	401		
1989-04-29 ltfssini.dtx v1.0f			
General: Corrections to L <sup>A</sup> T <sub>E</sub> X tabular env. added. . . . .	509		
1989-05-01 ltfssbas.dtx v1.0j			
General: Default for \baselinestretch added. . . . .	401		
1989-05-22 ltfssbas.dtx v1.0k			
General: Lines longer than 72 characters folded. . . . .	401		
1989-05-22 ltfssini.dtx v1.0g			
General: Lines shortened to 72 characters . . . . .	509		
1989-09-14 ltfssbas.dtx v1.0m			
General: Global replacement: \group to \mathgroup . . . . .	401		
\mathversion: Corrected typo: \endcsname to \endcsmame. . . . .	412		
1989-11-07 ltfssini.dtx v1.0i			
General: All family, series, and shape names abbreviated. . . . .	509		
1989-11-08 ltfssbas.dtx v1.0o			
General: First parameter of \define@mathalphabet and \define@mathgroup changed from string to control sequence. . . . .	401		
1989-11-14 ltfssbas.dtx v1.0p			
\math@version: Math version prefix 'mv@' added. . . . .	412		
1989-11-19 ltfssbas.dtx v1.0q			
\define@newfont: Group added. . . . .	414		
\wrong@fontshape: Instead of calling \family\default@family, etc. we directly set \f@family, etc. . . . .	419		
1989-11-22 ltfssbas.dtx v1.0r			
\math@version: \def → \edef for \math@version. . . . .	412		
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. . . . .	401		
extra// → + in \extra@def. . . . .	401		

1989-11-26 ltfssbas.dtx v1.0t \select@group: \bgroup/\egroup changed to \begingroup/\endgroup to avoid empty Ord atom on math list. . .	421	1990-01-25 ltfssini.dtx v1.1e \nfss@text: Macro added. . . . .	532
1989-12-02 ltfssini.dtx v1.1b General: \rmmath renamed to \mathrm . . . . .	509	1990-01-27 ltfssbas.dtx v1.2d \DeclarePreloadSizes: Font identifier set to \relax. . . . .	407
1989-12-03 ltfssini.dtx v1.1c General: Some internal macros renamed to make them inaccessible. . . . .	509	1990-01-28 ltfssbas.dtx v1.2e \mathgroup: \newfam let to \new@mathgroup. . . . .	401
1989-12-05 ltfssbas.dtx v1.0u \addto@hook: \addto@hook added. . .	426	1990-01-28 ltfssbas.dtx v1.2f \define@newfont: Added call to \curr@fontshape macro to allow substitution. . . . .	414
1989-12-05 lfsstrc.dtx v1.0u fam.dtx \every@math@size: Hook \every@size added. . . . .	458	\wrong@fontshape: Warning message slightly changed. . . . .	419
1989-12-13 lfsstrc.dtx v1.0f \use@mathgroup: \expandafter added before final \fi. . . . .	461	1990-01-28 ltfssini.dtx v1.2b \em: Call to \nomath added. . . . .	529
1989-12-16 ltfssbas.dtx v1.1a \select@group: \relax in front added. . . . .	421	1990-02-08 ltfssini.dtx v1.1g General: Protected the commands \fam, \series, \shape, \size, \selectfont, and \mathversion. . . . .	509
Now four arguments. . . . .	421	1990-02-16 ltfssbas.dtx v1.2g General: Support for changes of \baselineskip without changing the size. . . . .	401
Redefinition of alphabet now simpler. . . . .	422	\mathversion: \nomath added. . . . .	412
Usage of '=' macro added. . . . .	422	1990-02-18 lfsstrc.dtx v1.0j \selectfont: Redefine unprotected version \p@selectfont instead of \selectfont. . . . .	453
1989-12-16 lfsstrc.dtx v1.1a \selectfont: Changed order of calls. . . . .	453	1990-03-14 lfsstrc.dtx v1.0k General: Added code for TeX3. . . . .	449
\use@mathgroup: Redefinition of alphabet now simpler. . . . .	460	\extract@font: Added code for TeX3. . . . .	452
Usage of '=' macro added. . . . .	460	1990-03-30 ltfssbas.dtx v1.2h \math@egroup: Changed to have one arg. . . . .	424
1990-01-18 lfsstrc.dtx v1.0h General: \tracingfonts meaning changed. . . . .	449	1990-03-30 lfsstrc.dtx v1.2h \use@mathgroup: Third argument removed (see \math@egroup). . . . .	460
1990-01-20 ltfssbas.dtx v1.2a \math@bgroup: Def. placed in this file. . . . .	424	1990-04-01 ltfssbas.dtx v1.2i General: Code added from tracefn.ttx. . . . .	401
\math@egroup: Def. placed in this file. . . . .	424	Support for TeX3. . . . .	401
\select@group: Def for alph id changed. . . . .	422	1990-04-01 lfsstrc.dtx v1.0l General: Part of code moved to fam.dtx. . . . .	449
1990-01-21 ltfssbas.dtx v1.2b \select@group: Code moved to \use@mathgroup. . . . .	422	\tracingfonts: Check if \tracingfonts already defined. . . . .	450
1990-01-21 lfsstrc.dtx v1.2b \use@mathgroup: Macro added to allow cleaner interface. . . . .	460	1990-04-01 lfsstrc.dtx v1.0o \tracingfonts: Check if \tracingfonts defined removed again. . . . .	450
1990-01-23 ltfssbas.dtx v1.2c General: \no@version@warning renamed to \no@alphabet@error. . . . .	401		
Macro \no@alphabet@help added . . . . .	401		
\no@alphabet@error: Changed to error call . . . . .	401		

1990-04-02 ltfssini.dtx v1.1i		1991-08-14 ltpictur.dtx LaTeX2.09	
General: \input of files now handled by docstrip. . . . .	509	General: (RmS) inserted extra braces around entry for NFSS . . . . .	706
1990-04-05 lfsstrc.dtx v1.0m		1991-08-14 ltthm.dtx LaTeX2.09	
\selectfont: Call \tracingon only if \tracingfonts greater than 3. . . . .	453	\endtheorem: Moved \itshape after \item to make it work with NFSS . . . . .	736
1990-05-05 lfsstrc.dtx v1.0n		1991-08-26 ltfssini.dtx v1.1n	
\selectfont: \tracingon with new syntax. . . . .	453	\reset@font: Macro introduced . . . . .	532
1990-06-23 ltfssini.dtx v1.1k		1991-08-26 ltmiscen.dtx LaTeX2.09	
\nfss@text: Changed to \mbox. . . . .	532	\overline: \@par added . . . . .	623
1990-06-24 ltfssbas.dtx v1.2j		1991-08-26 ltpictur.dtx LaTeX2.09	
\DeclarePreloadSizes: Missing percent added. . . . .	407	\endpicture: (RmS & FMi) extra boxing level around \picbox to guard against unboxing in math mode (proposed by John Hobby)	704
1990-06-24 lfsstrc.dtx v1.0o		1991-08-26 ltplain.dtx LaTeX209	
\baselinestretch: Moved to tracefntr.dtx. . . . .	458	\tracingall: Added \errorcontextlines=\maxdimen, suggested by J. Schrod . . . . .	32
\getanddefine@fonts: \Adding tracing code. . . . .	462	1991-09-29 ltboxes.dtx LaTeX2.09	
\Macro moved from fam.dtx. . . . .	461	\cmpfootnotetext: (RmS) added \reset@font . . . . .	672
Adding debug code. . . . .	461	1991-09-29 ltfloat.dtx LaTeX2.09	
\use@mathgroup: Tracing code added. . . . .	461	\footnotetext: (RmS) added \reset@font . . . . .	769
1990-06-30 ltfssbas.dtx v1.2l		1991-09-29 ltmath.dtx LaTeX2.09	
\showhyphens: Macro added. . . . .	424	\eqnum: RmS: \reset@font added. . . . .	638
1990-06-30 lfsstrc.dtx v1.0p		1991-09-29 ltsect.dtx LaTeX2.09	
\use@mathgroup: Added \relax after math group number. . . . .	461	\dottedtocline: (RmS) added \reset@font for page number . . . . .	748
1990-07-07 lfsstrc.dtx v1.0q		1991-10-17 ltcntrl.dtx LaTeX209	
\getanddefine@fonts: Group number added to tracing. . . . .	462	\ctfor: (Rms) \xdef replaced by \def (See FMi's array.doc) . . . . .	274
\math@egroup: Tracing code added. . . . .	461	1991-10-25 ltbibl.dtx LaTeX2.09	
\use@mathgroup: Group number added to tracing. . . . .	461	\citex: added \reset@font, suggested by Bernd Raichle. . . . .	777
1990-08-27 lfsstrc.dtx 1.0r		1991-11-01 ltfloat.dtx LaTeX2.09	
\type@restoreinfo: Some extra tracing info. . . . .	457	\footnote: (RmS) Added \let\protect\noexpand in \footnote, \footnotemark, and \footnotetext, since \xdef is used . . . . .	769
1990-08-27 lfsstrc.dtx v1.0r		1991-11-04 ltlists.dtx LaTeX2.09	
\getanddefine@fonts: Correcting missing name after \tracingon. . . . .	461	\makelabel: (RmS) added default definition for \makelabel, to produce an error message. . . . .	657
1991-03-28 ltfssini.dtx v1.1m		1991-11-04 ltplain.dtx RmS	
\copyright: Extra braces added. . . . .	532	General: Removed \itemitem since never needed/useful with L <sup>A</sup> T <sub>E</sub> X. . . . .	30
1991-03-30 ltfssini.dtx v1.2g		1991-11-06 ltbibl.dtx LaTeX2.09	
\newfont: Definition added. . . . .	531	\citex: added code to remove a leading blank . . . . .	777
\symbol: Definition added. . . . .	531		
1991-07-24 ltmiscen.dtx LaTeX2.09			
\verbatim: Added \penalty\interlinepenalty to definition of \par so that \samepage works . . . . .	623		
1991-08-14 ltmath.dtx LaTeX2.09			
\cases: (RmS) inserted extra braces around entry for NFSS . . . . .	633		

1991-11-13	ltbibl.dtx	LaTeX2.09		avoid conflicts with other channels allocated by \newread . . . . .	75
	\@bibitem:	Changed counter enumi to enumiv, as it says in the comment above . . . . .	777		
1991-11-21	ltfssini.dtx	v1.10		\@xmpar: (RmS) added \global\@ignorefalse . . . . .	764
	\reset@font:	Added extra braces for robustness. . . . .	532	\end@float: (RmS) changed \@esphack to \@Espack . . . . .	758
		Changed to protected version of macro. . . . .	532		
1991-11-22	ltfloat.dtx	LaTeX2.09		1992-03-18 ltlstlist.dtx 0.0 \trivlist: RmS: added \@nmbrrlistfalse . . . . .	653
	\footnote:	(RmS) Added \let\protect\noexpand in \xfootnote, \xfootnotemark, and \xfootnotetext . . . . .	769	1992-03-18 ltmiscen.dtx LaTeX2.09 \begin: Changed \@ignoretrue to \@ignorefalse (as documented)	615
1991-11-22	ltlists.dtx	LaTeX2.09		1992-03-21 ltfsini.dtx v1.2d General: Renamed \text to \nfss@text to make it internal.	509
	\@item:	(RmS) Changed second call to \makelabel to \unhbox\@tempboxa. Avoids problems with side effects in \makelabel and is more efficient.	657	1992-05-12 ltfsbas.dtx v1.3c \extract@alph@from@version: Macro added. . . . .	423
1991-11-27	ltfssbas.dtx	v1.3a		\select@group: Added call to \extract@alph@from@version. . . . .	422
	General:	All \family, \shape etc. renamed to \fontfamily etc. . . .	401	1992-07-26 ltfsbas.dtx v1.9a \curr@fontshape: . . . . .	414
1991-11-27	ltfsini.dtx	v1.2a		\DeclareFontShape: Introduced \DeclareFontShape . . . . .	402
	General:	All \family, \shape etc. renamed to \fontfamily etc. . . .	509	\define@newfont: . . . . .	414
1992-01-06	ltfsini.dtx	v1.2c		\math@fonts: . . . . .	421
	General:	added slitex code . . . . .	509	\select@group: . . . . .	422
1992-01-10	ltbibl.dtx	LaTeX2.09		\split@name: Added splitting into \f@encoding. . . . .	414
	\@bibitem:	Changed \c@enumiv to \value of \@listctr . . . . .	777	\wrong@fontshape: . . . . .	419
1992-01-10	ltmath.dtx	LaTeX2.09		1992-07-26 lfsstrc.dtx v2.0b \s@fct@: . . . . .	470
	equation:	RmS: put \hbox around \eqnnum to typeset the equation number in text mode (as in the eqnarray env.) . . . . .	638	\s@fct@sub: documentation fixes . . . . .	471
1992-01-10	ltthm.dtx	LaTeX2.09		\selectfont: . . . . .	454
	\@othm:	(RmS) Check for existence of theorem environment . . . . .	735	\try@simple@size: . . . . .	464
1992-01-14	ltbibl.dtx	LaTeX2.09		\try@size@range: . . . . .	468
	\@biblabel:	removed \hfill . . . . .	780	\use@mathgroup: . . . . .	461
1992-01-14	ltsect.dtx	0.0		1992-08-14 ltbibl.dtx LaTeX2.09 \@citex: added missing argument braces around \hbox, found by Ed Sznyter . . . . .	777
	\@starttoc:	(RmS) added \immediate to \openout as all \write commands are also executed \immediate . . . . .	746	1992-08-14 ltboxes.dtx LaTeX209 \endminipage: (RmS) replaced \vskip-\lastskip by \unskip (proposed by FMi) . . . . .	671
1992-02-26	ltbibl.dtx	LaTeX2.09		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) . . . . .	777
	\@lbibitem:	Added \hfill to restore left-alignment of bibliography labels in alpha style . . . . .	777	1992-08-19 ltsect.dtx 0.0 \@xsect: (RmS) corrected bug: stretch and shrink in argument to \hskip	
1992-03-18	ltdefns.dtx	LaTeX209			
	General:	(RMS) changed input channel from 0 to \inputcheck to			

previously not negated . . . . .	742	\footnote: (RmS) Changed all to ‘def’protect’noexpand’protect’noexpand . . . . .	769
1992-08-19 ltthm.dtx LaTeX2.09			
\@othm: (RmS) Changed error message to complain about undefined counter . . . . .	735	\hexnumber@: Make it accept counters. . . . .	532
1992-08-20 ltfssini.dtx v1.4b			
\@setsizE: Added \@currsize. . . . .	531	1993-03-08 preload.dtx v2.0b General: Added 12pt preloads . . . . .	558
1992-08-24 ltdefns.dtx LaTeX209			
\@ifnextchar: (Rms) \@ifnextchar didn't work if its first argument was an equal sign. . . . .	97	1993-03-18 ltfssbas.dtx v2.0c General: Changed all \@tempdima in \@tempdimb to avoid killing \numberline . . . . .	401
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. . . . .	615	1993-03-18 lfsstrc.dtx v2.1b General: Changed all \@tempdima in \@tempdimb to avoid killing \numberline . . . . .	449
\verb: Changed \verb and \@sverb to work correctly in math mode . . . . .	628	Changed all \@tempdimb in \@tempdimx to avoid killing \numberline . . . . .	449
1992-08-25 ltsect.dtx LaTeX2.09			
\@sect: (FMi) replaced explicit setting of \@svsec by call to \@seccntformat . . . . .	741	1993-03-18 lfsstrc.dtx v2.1c \DeclareSizeFunction: Added all args to avoid blanks problems . . . . .	467
1992-09-18 ltlists.dtx LaTeX2.09			
\item: (RmS) Added warning if \item is used in math mode . . . . .	655	1993-04-09 lterror.dtx v1.0e \@latexerr: Mention The Companion . . . . .	281
1992-09-18 ltab.dtx LaTeX2.09			
\@array: Changed \par to \empty to avoid starting new row e.g. after \hline . . . . .	689	1993-04-11 lterror.dtx v1.0f \@latexerr: Remove setting of errorcontextlines . . . . .	281
1992-09-19 lfsstrc.dtx v2.0c			
\try@simple@size: . . . . .	464	1993-05-05 lftntcmd.dtx v2.0b General: Removed all LaTeX related cmds . . . . .	561
1992-09-21 ltfssini.dtx v1.4d			
\@not@math@alphabet: Macro defined. . . . .	530	1993-05-16 ltfssbas.dtx v2.0e \showhyphens: Use \reset@font . . . . .	424
1992-09-22 ltfssbas.dtx v1.91a			
General: Introduced \tf@size for math size. . . . .	401	1993-07-16 lfsstrc.dtx v2.1h General: Changed layout of info messages . . . . .	449
1992-09-22 lfsstrc.dtx v2.1a			
\getanddefine@fonts: Introduced \tf@size for math size. . . . .	462	1993-07-17 ltoutenc.dtx 1.0d General: changed \catcoding @ . . . . .	348
1992-11-13 ltfssini.dtx v?			
\hexnumber@: Made expandable. . . . .	532	1993-08-03 ltmiscen.dtx LaTeX2.09 \enddocument: Changed redefinition of \global to redefinition of \@setckpt. . . . .	609
1992-11-23 ltcounds.dtx LaTeX209			
\stepcounter: Replaced {} in \stepcounter by \begingroup \endgroup to avoid adding an empty ord in math mode . . . . .	393	1993-08-05 ltpictur.dtx LaTeX2.09 \circle: (RMS) Added error message if \circle is used in math mode. . . . .	726
1992-11-26 ltboxes.dtx LaTeX2.09			
\@mpfootnotetext: (RmS) added protection for \edef . . . . .	672	1993-08-05 ltsect.dtx LaTeX2.09 \@sect: (RmS) Made sure that \protect works correctly in expansion of \the counter . . . . .	741
1992-11-26 lfloat.dtx LaTeX2.09			
\@footnotetext: (RmS) added protection for \edef . . . . .	769	1993-08-05 ltspace.dtx LaTeX2e \@hspacE: (RmS) Removed superfluous \leavevmode in \@hspacE and \@hspacEr, as suggested by CAR. . . . .	321

1993-08-05 lttab.dtx latex2e	\everycr . . . . .	706
	\tabular*: Replaced	
	\expandafter\def by \cnamedef.	688
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. . . . .	777
1993-08-13 ltoutenc.dtx 1.0f	General: Protected against active @	
	sign. . . . .	348
1993-08-13 preload.dtx v2.0c	General: Added \relax at end of font	
	names. . . . .	559
1993-08-16 ltoutenc.dtx 1.0g	General: Needs space after \string	348
1993-08-18 ltfssdcl.dtx v2.0e	\new@mathversion: Exchanged names	
	of encodings in warning message of	
	\SetSymbolFont. . . . .	490
1993-09-02 lfsstrc.dtx v2.1i	General: Corrected name of sgen size	
	function. . . . .	449
1993-09-03 ltmiscen.dtx LaTeX2.09	\verb@nolig@list: Replaced	
	\@noligs by extensible list . . . . .	628
1993-09-07 ltmiscen.dtx LaTeX2.09	\verb@balance@group: (RmS)	
	Changed definition of \verb so	
	that it detects a missing second	
	delimiter. . . . .	627
1993-09-08 ltmiscen.dtx LaTeX2.09	\enddocument: Added warning in case	
	of undefined references. . . . .	609
1993-09-15 ltfsbas.dtx v2.0g	\DeclareFontEncoding: Corrected:	
	\default@T to \default@M. . . . .	405
1993-09-15 lfsstrc.dtx v2.1j	General: Corrected spelling of	
	\noxpand. . . . .	449
1993-09-19 lterror.dtx LaTeX2.09	\@invalidchar: (RmS) Error message	
	for invalid input characters. . . . .	285
1993-11-02 ltmath.dtx LaTeX2.09	General: RmS: Corrected description	
	of \eqnse1, moved \eqnse1	
	accordingly and removed extra	
	\tabskip assignment. . . . .	638
1993-11-03 ltmath.dtx LaTeX2e	General: RmS: Initialized \everycr to	
	empty . . . . .	638
1993-11-03 ltpictur.dtx LaTeX2.09	General: (RmS) changed \halign to	
	\ialign to initialize \tabskip and	
	\everycr . . . . .	706
	\normalfont: Macro added . . . . .	532
1993-11-11 ltfssini.dtx v2.1a	\currext: Name changed from	
	\currextension . . . . .	790
1993-11-11 lfsstrc.dtx v2.2a	\resetoptions: macro added . . . . .	816
	\AtEndDocument: Included extension	
	in the generated macro name for	
	package and class hooks. . . . .	816
\documentstyle: Added	\RequirePackage	
	\@unusedoptionlist stuff. . . . .	805
\load@onefilewithoptions: Moved	resetting of \default@ds, \ds@ and	
	\@declaredoptions here, from the	
	end of \ProcessOptions. . . . .	810
\NeedsTeXFormat: made more robust	for alternative syntax for other	
	formats. . . . .	807
\ProcessOptions*: Optimize ‘empty	option’ code. . . . .	802
	Stop adding the global option list	
	inside class files. . . . .	802
1993-11-14 ltdefns.dtx v0.2a	\g@addto@macro: Made global . . . . .	102
1993-11-15 ltclass.dtx v0.2b	\documentstyle: Modified to match	
	\ProcessOption*. . . . .	805
	\ProcessOptions*: Star form added. . . . .	802
1993-11-17 ltclass.dtx v0.2c	\@fileswith@ptions: Macro added	815
	\@badrequireerror: Macro added . . . . .	817
	\@twoloadclasserror: Macro added	818
\CurrentOption: Name changed from	\@curroption . . . . .	790
	\DeclareOption*: Error checking	
	added . . . . .	800
\load@onefilewithoptions: Added	trap for two \LoadClass	
	commands. . . . .	812
\NeedsTeXFormat: Name changed from	\NeedsFormat . . . . .	807
	\ProcessOptions*: restoring	
	\fileswith@ptions added. . . . .	802
1993-11-18 ltclass.dtx v0.2d	\documentstyle: Modified	
	\RequirePackage stuff. . . . .	805
\ExecuteOptions: Use	\CurrentOption not \reserved@a	804

\NeedsTeXFormat: \fmtname		\newcommand: Macro reimplemented
\fmtversion not \C... . . . . .	807	and extended . . . . .
1993-11-21 lfiles.dtx LaTeX2e		\renewcommand: Macro reimplemented
\@missingfileerror: Stop infinite		and extended . . . . .
looping on \Cer@ext . . . . .	341	\renewenvironment: Macro
1993-11-21 ltmiscen.dtx v0.9a		reimplemented and extended . . . . .
\@verbatim: use \verb@font		\two@digits: Macro added . . . . .
instead of \tt . . . . .	624	1993-11-23 ltoutput.dtx v0.1a
\verb: Use \verb@font instead of		\paperheight: Register added . . . . .
\tt. . . . .	628	\paperwidth: Register added . . . . .
\verb@font: Macro added . . . . .	624	1993-11-23 ltoutput.dtx v0.1c
1993-11-22 ltclass.dtx v0.2f		\enlargepage: Command added . . . . .
\@fileswithoptions: Made the		\kludgeins: Insert added . . . . .
default [] not [\@unknownversion] . . . . .	808	\makecol: Command changed . . . . .
\@ifl@ter: Added //00 so parsing		\specialoutput: Command changed . . . . .
never produces a runaway		\enlargethispage*: Commands
argument. . . . .	795	added . . . . .
General: \@unknownversion removed . . . . .	785	1993-11-24 lftntcmd.dtx v2.1a
\load@onefilewithoptions: Made the		\maybe@ic@: Use \t@st@ic . . . . .
initial version [] not		\t@st@ic: Macro added . . . . .
[\@unknownversion] . . . . .	810	1993-11-24 ltfssini.dtx v2.1a
1993-11-22 ltdefns.dtx LaTeX2e		General: Removed \xpt stuff . . . . .
\@minus: Macro added . . . . .	74	1993-11-24 ltlogos.dtx LaTeX2e
\@plus: Macro added . . . . .	74	\LaTeX: Macro changed . . . . .
\CheckCommand: Macro added . . . . .	81	1993-11-28 ltclass.dtx v0.2h
\providecommand: Macro added . . . . .	81	\@twoclasseserror: Macro added . . . . .
1993-11-22 lterror.dtx LaTeX2e		General: Assorted commands now in
\c@errorcontextlines: Macro added . . . . .	281	the kernel removed. . . . .
1993-11-22 lfiles.dtx LaTeX2e		Directory syntax checking moved to
\listfiles: Removed checking for		dircheck.dtx . . . . .
\@unknownversion . . . . .	343	Primitive filenames now terminated
1993-11-22 llength.dtx LaTeX2e		by space not \relax. . . . .
\@settodim: Macro added . . . . .	399	\endfilecontents: Don't globally
\@settopoint: Macro added . . . . .	400	allocate a write stream (always use
\@settodepth: Macro added . . . . .	399	15) . . . . .
\@settoheight: Macro added . . . . .	399	1993-11-28 ltfiles.dtx LaTeX2e
1993-11-22 ltlogos.dtx LaTeX2e		\@missingfileerror: Use filename
\LaTeXe: Macro added . . . . .	323	parser from dircheck . . . . .
1993-11-23 ltclass.dtx v0.2g		1993-11-29 ltoutput.dtx v1.0b
\@use@option: Name changed from		\@makecol: \@makespecialcolbox
\@executeoption . . . . .	804	added . . . . .
General: Various macros now moved		\@makespecialcolbox: Command
to latex.tex. . . . .	789	added . . . . .
Warnings and errors now directly		1993-11-29 lplain.dtx LaTeX2e
coded. . . . .	789	General: All accents in decimals;
1993-11-23 ltdefns.dtx LaTeX2e		suggested by Paul Taylor . . . . .
\@argdef: Macro added . . . . .	77	1993-11-30 ltoutput.dtx v1.0c
\@ifundefined: Redefined to remove a		\f@tracemessage: Commands added . . . . .
trailing \fi . . . . .	96	1993-12-01 fontdef.dtx v2.1a
\@newcommand: Macro added . . . . .	77	General: Update for LaTeX2e . . . . .
\@newenv: Macro interface changed . . . . .	80	1993-12-01 ltoutput.dtx v1.0e
\@xargdef: Macro interface changed . . . . .	77	\@reinserts: Command added . . . . .
\@yargd@f: Avoid \C?@? token . . . . .	78	1993-12-03 ltboxes.dtx v0.1a
Macro interface changed . . . . .	78	\@argsbox: macro removed . . . . .

\@begin@tempboxa: macro added . . .	662	1993-12-05 ltfloat.dtx LaTeXe
\@end@tempboxa: macro added . . .	662	\@dblfloataplacement: Command changed . . . . .
\@iirsbox: redefined to support \height . . . . .	674	760
\@imakebox: macro modified . . . . .	662	\@xfloat: Command changed . . . . .
\@iirsbox: redefined to support \height . . . . .	674	754
\@isavebox: color support . . . . .	665	1993-12-05 ltoutput.dtx v1.0f
extra group . . . . .	665	\@addtobot: Command changed . . . . .
\@isavepicbox: extra group . . . . .	665	922
\@makebox: default changed from x to c . . . . .	662	\@addtocurcol: Command changed . . . . .
\@makepicbox: macro modified . . . . .	663	924
\@savebox: default c not x . . . . .	665	\@addtobblcol: Command changed . . . . .
\bm@b: macros added . . . . .	662	935
\endlrbox: macro added . . . . .	666	\@addtonextcol: Command changed . . . . .
\fbox: extra group . . . . .	666	931
\lrbbox: color support . . . . .	665	\@addtotoporbot: Command changed . . . . .
macro added . . . . .	665	923
\makebox: modified . . . . .	661	\@boxfpsbit: Command added . . . . .
\mbox: extra group . . . . .	662	947
\minipage: Redefined to support extra optional arguments . . . . .	670	\@flcheckspace: Command added . . . . .
\newsavebox: Pass the whole of arg 1 to \ifdefinable . . . . .	664	949
\parbox: Redefined to support extra optional arguments . . . . .	668	\@flsetnum: Command added . . . . .
\raisebox: redefined to support \height . . . . .	674	948
\sbox: color support . . . . .	665	\@flsettextmin: Command added . . . . .
extra group . . . . .	665	949
\set@color: color support . . . . .	664	\@flstop: Commands added . . . . .
macro added . . . . .	664	945
1993-12-03 ltclass.dtx v0.2i		\@flupdates: Command added . . . . .
\@cls@pkg: Name changed to avoid clash with output routine. . . . .	817	950
General: \@onlypreamble: Many commands declared. . . . .	789	\@fpsadddefault: Command added . . . . .
Removed obsolete \@documentclass . . . . .	789	\@getfpsbit: Command added . . . . .
1993-12-03 lterror.dtx v1.0b		\@opcol: Command changed . . . . .
\@latexerr: Set \c@errorcontextlines to -1 . . .	281	908
1993-12-03 ltfsini.dtx v2.1a		\@outputpage: Command changed . . . . .
General: update for LaTeXe . . . . .	509	912
1993-12-04 ltfilehook.dtx v0.9b		\@resethfps: Command added . . . . .
\unqu@tefilef@und: Macro added . . . . .	845	948
1993-12-04 ltfiles.dtx v0.9b		\@setfloattypecounts: Command added . . . . .
\@iinput: Macro reimplemented . . . . .	340	946
\@input: Macro reimplemented . . . . .	341	\@setfpsbit: Command added . . . . .
\IfFileExists@: Macro added . . . . .	338	947
\input: Macro reimplemented . . . . .	340	\@shipoutsetup: Command added . . . . .
1993-12-06 ltclass.dtx v0.2k		912
\ExecuteOptions: Preserve \CurrentOption . . . . .		902
1993-12-06 ltoutput.dtx v1.0f		\@startcolumn: Command changed . . . . .
\@specialoutput: Unboxing of 255 added to rescue writes . . . . .		918
1993-12-06 ltoutput.dtx v1.0g		\@startdblcolumn: Command changed . . . . .
\@topnewpage: \@floatplacement placement bug fixed . . . . .		918
1993-12-07 ltclass.dtx v0.2l		\@testfp: Command added . . . . .
\ProvidesFile: Macro added . . . . .		947
1993-12-07 ltclass.dtx v0.2m		\@textfloatsheight: Commands added . . . . .
\load@onefilewithoptions: Reset \CurrentOption . . . . .		946
1993-12-07 ltoutenc.dtx 1.1		\@topnewpage: Commands changed . . . . .
General: Protected all special characters with \string . . . . .		900
1993-12-07 ltoutenc.dtx 1.1		\@tryfcolumn: Command changed . . . . .
General: Protected all special characters with \string . . . . .		919
1993-12-07 ltoutput.dtx v1.0f		\@writesetup: \@startpagehook added . . . . .
\@output: Command changed . . . . .		912
1993-12-06 ltclass.dtx v0.2k		\@outputpage: Command changed . . . . .
\ExecuteOptions: Preserve \CurrentOption . . . . .		902
1993-12-06 ltoutput.dtx v1.0f		\@specialoutput: Unboxing of 255 added to rescue writes . . . . .
1993-12-06 ltoutput.dtx v1.0g		902
\@topnewpage: \@floatplacement placement bug fixed . . . . .		900
1993-12-07 ltclass.dtx v0.2l		\@startcolumn: Command changed . . . . .
\ProvidesFile: Macro added . . . . .		799
1993-12-07 ltclass.dtx v0.2m		\@startdblcolumn: Command changed . . . . .
\load@onefilewithoptions: Reset \CurrentOption . . . . .		810
1993-12-07 ltoutenc.dtx 1.1		\@topnewpage: Commands changed . . . . .
General: Protected all special characters with \string . . . . .		900
1993-12-07 ltoutenc.dtx 1.1		\@tryfcolumn: Command changed . . . . .
General: Protected all special characters with \string . . . . .		919

1993-12-07 ltoutenc.dtx v1.1		1993-12-11 ltmath.dtx v0.9g	
General: Made all character numbers decimal. ....	345	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. ....	638
Removed a lot of equal signs and the like. ....	345		
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) ....	662	General: Corrected for t1enc, math.	345
<code>\@irsbox</code> : fix typo ....	674	1993-12-11 ltsect.dtx LaTeXe	
<code>\@parboxto</code> : <code>\endgraf</code> added due to extra group in <code>\@begin@tempboxa</code> ....	669	<code>\author</code> : Added default ....	737
<code>\lrbbox</code> : move <code>\endpfalse</code> out of the inner group ....	665	<code>\title</code> : Added default ....	737
1993-12-08 ltfntcmd.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 ....	569	<code>\@setref</code> : Macro added ....	605
1993-12-08 ltlists.dtx LaTeXe		<code>\pageref</code> : Macro reimplemented ...	605
<code>\@item</code> : use <code>\sbox</code> to support colour ....	656	<code>\ref</code> : Macro reimplemented ....	605
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 ....	311	<code>\@cflb</code> : <code>boxmaxdepth</code> setting moved defs changed to lets ....	916
<code>\@vbsphack</code> : Command added ....	314	<code>\@cflt</code> : name changed ....	916
1993-12-09 ltboxes.dtx v0.1c		<code>\@doclearpage</code> : defs changed to lets ....	907, 908
<code>\@irsbox</code> : fix another typo ....	674	<code>\@makewcol</code> : defs changed to lets ...	909
1993-12-09 ltclass.dtx v0.2n		<code>\@resethfps</code> : Warnings added: minimal ....	948
<code>\documentstyle</code> : input 209 compatibility file. ....	805	<code>\@startdblcolumn</code> : defs changed to lets ....	918, 919
1993-12-09 ltfiles.dtx v0.9e		<code>\@topnewpage</code> : braces removed ....	900
<code>\document</code> : Hook added ....	326	<code>\@tryfcolumn</code> : defs changed to lets ..	919
1993-12-09 ltmiscen.dtx v0.9e		<code>\f@tracemessage</code> : Commands changed ....	943
<code>\enddocument</code> : Hook added ....	609	1993-12-13 ltclass.dtx v0.2o	
1993-12-10 ltoutenc.dtx v1.2		General: Removed setting <code>\errorcontextlines</code> (now in latex.tex) ....	789
General: Added source code for t1enc.sty. ....	345	<code>\documentstyle</code> : compatibility file now latex209.sty. ....	805
1993-12-11 ltfntcmd.dtx v3.0a		<code>\usepackage</code> : Fixed error handling ..	807
General: Complete reworking of all text commands, using just one creator function ....	561	1993-12-13 ltdirchk.dtx v0.2a	
italic correction now put in front of penalty before glue ....	561	General: on the 'docstrip' pass, do not check openin path ....	10
newcommands replaced by defs ..	561	<code>\IfFileExists</code> : Removed interactive prompting for current directory syntax ....	10
newfontswitch command corrected and changed ....	561	<code>\strip@prefix</code> : modified, name changed from <code>\stripmeaning</code> . ...	5
<code>\DeclareTextFontCommand</code> : Macro changed ....	563	1993-12-13 ltlists.dtx latex2e	
<code>\emph</code> : Macro changed ....	564	<code>\trivlist</code> : Initialised <code>\@itemlabel</code>	653
<code>\fix@penalty</code> : Macro added ....	567	1993-12-13 ltmiscen.dtx v0.9h	
<code>\maybe@ic</code> : Macro name changed ..	566	<code>\@noligs</code> : Readded <code>\@noligs</code> ....	628
<code>\maybe@ic@</code> : Macro and name changed ....	566	<code>\@verbatim</code> : Readded <code>\@noligs</code> ... Removed optional argument of <code>\item</code> ....	623
<code>\sw@slant</code> : Macro changed ....	567	<code>\center</code> : Removed optional argument of <code>\item</code> ....	621
<code>\textup</code> : Macros changed ....	564		

<b>flushleft:</b> Removed optional argument of <code>\item</code> . . . . .	622	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 <code>\ProvidesPackage</code> commands. Added <code>XXXenc</code> . . . . .	348
<b>flushright:</b> Removed optional argument of <code>\item</code> . . . . .	622		
1993-12-13 ltoutenc.dtx v1.2b			
General: Corrected file name in driver code. . . . .	345		
1993-12-13 lttab.dtx latex2e			
<code>\tabbing:</code> Removed optional argument of <code>\item</code> . . . . .	683		
1993-12-14 ltoutput.dtx v1.0i			
General: Section added to declare all parameters . . . . .	955	Made Rokicki's encoding a proper encoding scheme rather than a variant of OT1. . . . .	345
1993-12-15 ltboxes.dtx v0.1d			
<code>\@iminipage:</code> Changed default from 'c' to 's' . . . . .	671		
<code>\@iparbox:</code> Changed default from 'c' to 's' . . . . .	668		
<code>\minipage:</code> Changed default from 'c' to 's' . . . . .	670		
extra space removed. . . . .	670		
<code>\parbox:</code> Changed default from 'c' to 's' . . . . .	668		
1993-12-15 ltclass.dtx v0.2p			
General: Removed extra 's' from <code>\@@warnings</code> . . . . .	789		
1993-12-16 ltlogos.dtx LaTeX2e			
<code>\LaTeXe:</code> Extended logo by DPC . . . . .	323		
1993-12-16 ltmath.dtx v0.9i			
<code>\@eqncr:</code> use <code>\refstepcounter</code> instead of shortcut . . . . .	640		
General: use <code>\refstepcounter</code> instead of shortcut . . . . .	638		
1993-12-16 ltmiscen.dtx v0.9i			
General: <code>\literal</code> added . . . . .	628		
1993-12-16 ltpage.dtx LaTeX2e			
<code>\mark:</code> Init <code>\mark</code> at begin document . . . . .	783		
1993-12-16 ltspace.dtx LaTeX2e			
<code>\@bsphack:</code> Corrected optimisation :-)	311		
1993-12-16 lttab.dtx latex2e			
<code>\xhline:</code> Measure from middle of vertical rules . . . . .	698		
1993-12-17 ltclass.dtx v0.2q			
<code>\@documentclasshook:</code> Macro added . . . . .	789		
<code>\@fileswithoptions:</code> Add <code>\@compatibility</code> hook . . . . .	808		
<code>\documentstyle:</code> Match Alan's new code. . . . .	805		
1993-12-17 ltoutenc.dtx 1.3			
General: Added this section . . . . .	349		
Removed all the hackery for use in <code>\DeclareFontEncoding</code> , and redid everything using <code>\DeclareTextFont</code> . . . . .	361, 363		
1993-12-17 ltoutenc.dtx v1.3a			
General: Replaced OT3 by XXX . . . . .	345		
1993-12-18 ltoutenc.dtx v1.3b			
General: Corrected typos. . . . .	345		
Replaced the missing last argument to <code>\DeclareFontEncoding</code> . . . . .	345		

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. . . . .	345	\@fileswithoptions: Modify to reduce parameter stack usage . . .	808
Rewrote for the new syntax of \EncodingSpecific. . . . .	345	General: Added many more \@onlypreamble commands . . . .	789
1993-12-18 ltoutenc.dtx v1.3d		Wrapped long lines to column 72	789
General: Some T1 stuff had drifted into the OT1 file. . . . .	345	\load@onefile@withoptions: Modify to reduce parameter stack usage	813
1993-12-18 ltpage.dtx LaTeXe		1994-01-17 ltfiles.dtx LaTeXe	
\sloppy: Added \emergencystretch	783	\listfiles: New Version, adds ‘tex’ if needed, and lines up columns .	343
1993-12-19 ltclass.dtx v0.2r		1994-01-17 ltfssbas.dtx v2.1a	
\endfilecontents: Different message when ignoring a file . . . . .	818	General: New math font setup . . .	401
1993-12-19 lftntcmd.dtx v3.0b		\curr@math@size: New math font setup . . . . .	413
General: \@pdef command added . .	561	\everydisplay: New math font setup	413
Added by ASAJ. . . . .	569	\everymath: New math font setup .	413
Made \@newfontswitch produce an error if command already exists, and added \@renewfontswitch, ASAJ . . . . .	561	\frozen@everydisplay: New math font setup . . . . .	413
Other tidying . . . . .	561	\frozen@everymath: New math font setup . . . . .	413
Some more tidying done . . . . .	561	\math@version: New math font setup	412
Untidying added, so this is now a TEMPORARY version. . . . .	561	1994-01-17 ltfssini.dtx v2.1e	
Wording changes by CAR. . . . .	569	\not@math@alphabet: Message changed . . . . .	530
\DeclareOldFontCommand: Corrected and tidied . . . . .	568	1994-01-17 lfsstrc.dtx v2.3a	
\DeclareTextFontCommand: Corrected and tidied . . . . .	563	General: New math font setup . . .	449
1993-12-19 ltspace.dtx LaTeXe		\check@mathfonts: New math font setup . . . . .	459
\@bsphack: There seem to be problems with selfmade birthday presents .	312	\glob@currscale: New math font setup	457
1993-12-20 ltdefs.dtx LaTeXe		\restglob@settings: New math font setup . . . . .	460
\@reargdef: Kept old version of \@reargdef, for array.sty . . . .	78	1994-01-18 ltbibl.dtx LaTeXe	
1993-12-20 ltfiles.dtx v0.9m		\bibliography: Use \@input@ so include files are listed. . . . .	778
\@obsoletefile: Added this command, removed @oldfilewarning . . . . .	343	1994-01-18 ltclass.dtx v0.2t	
1994-01-05 fontdef.dtx v2.1d		\@ifclassloaded: Fix typo \@pkgetension . . . . .	794
General: Removed nf prefix from file names. . . . .	539	1994-01-18 ltfilehook.dtx v0.9p	
1994-01-13 ltmath.dtx v0.9o		\unqu@tefilef@nd: New Definition	845
\@eqnacr: correcting 0.9i . . . .	640	1994-01-18 ltfiles.dtx v0.9p	
General: correcting 0.9i . . . .	638	\@iffileonpath: Macro added . . .	339
1994-01-14 ltdirchk.dtx v0.2d		\@input: do not use a different definition for \input@path . . . .	341
\IfFileExists: Close the texsys.aux output stream . . . . .	10	\@input@: Macro added . . . . .	341
1994-01-15 ltfiles.dtx v0.9o		\IfFileExists@: New Definition .	338
\document: move \@preamblecmds after document hook . . . . .	328	\includeonly: Use \@input@ so include files are listed. . . . .	332
		1994-01-18 ltfssini.dtx v2.1f	
		\not@math@alphabet: Message corrected . . . . .	530
		1994-01-18 ltmiscen.dtx v0.9p	
		\@verbatim: Add \global\@inlabelfalse . . . . .	623

Only add <code>\penalty</code> if in hmode . . . . .	623	<code>\restglb@settings</code> : Correct trace info placement . . . . .	460
1994-01-19 fontdef.dtx v2.1e			
General: Added missing setting for symbols in bold version. . . . .	544	<code>\nocorrlist</code> : Only ., used as default for cm fonts . . . . .	568
1994-01-19 ltdirchk.dtx v0.2e			
<code>\IfFileExists</code> : name changed from <code>\test</code> . . . . .	9	<code>\load@onefile@withoptions</code> : All options raise error if no <code>\ProcessOptions</code> appears . . . . .	813
<code>\input@path</code> : No longer check that an empty group is in the path . . . . .	10		
<code>\strip@prefix</code> : name changed from <code>\strip@meaning</code> , to match NFSS. . . . .	5		
1994-01-19 ltmath.dtx v1.0n classes			
<code>\mathindent</code> : Deferred setting of <code>\mathindent</code> . . . . .	641	<code>\g@addto@macro</code> : Use toks register to avoid ‘hash’ problems . . . . .	102
1994-01-20 ltdirchk.dtx v0.2f			
General: <code>\@copytexsys</code> and the <code>texsys.new</code> file removed . . . . .	8	<code>\document</code> : set <code>\@normalsize</code> or <code>\normalsize</code> if necessary . . . . .	327
Modify all of ltxcheck . . . . .	13		
<code>\IfFileExists</code> : <code>\@copytexsys</code> removed . . . . .	10		
1994-01-21 ltclass.dtx v0.2u			
<code>\documentstyle</code> : compatibility file now <code>latex209.def</code> . . . . .	805	General: <code>\@normalsize</code> no longer defined . . . . .	561
1994-01-21 ltdirchk.dtx v0.2g			
General: Improve documentation, reorganize docstrip module . . . . .	1	<code>\pagestyle</code> : (DPC) Modify to get nicer error message . . . . .	781
<code>\filename@parse</code> : Minor changes, and add Mac version () . . . . .	11	<code>\thispagestyle</code> : (DPC) Modify to get nicer error message . . . . .	782
<code>\today</code> : Name changed from <code>\stamp</code> , to save memory . . . . .	9		
1994-01-21 ltfloat.dtx LaTeX2e			
<code>\@xfloat</code> : Added missing percent characters. . . . .	754	<code>\load@onefile@withoptions</code> : Only run the hook and options check if the file was loaded. . . . .	813
1994-01-21 ltmiscen.dtx v0.9s			
<code>\verbatim@font</code> : Removed unnecessary category code hackery. . . . .	624	<code>\@makespecialcolbox</code> : correct mistakes in the documentation . . . . .	911
1994-01-24 ltdirchk.dtx v0.2h			
<code>\IfFileExists</code> : Stop testing once <code>texsys.aux</code> has been found . . . . .	9	<code>\@fileswithoptions</code> : Run <code>\@compatibility</code> on the first class to start (not the first to finish) . . . . .	808
1994-01-24 ltpage.dtx LaTeX2e			
<code>\pagestyle</code> : (DPC) Complain if <code>pagestyle</code> is undefined. . . . .	781	<code>\@ifclasswith</code> : Add extra ,s so ‘two’ is not matched with ‘twocolumn’ . . . . .	796
1994-01-25 ltdirchk.dtx v0.2i			
General: Protect against looping on <code>\@input</code> and <code>\@end</code> . . . . .	2	<code>\ProcessOptions*</code> : Add extra ,s so ‘two’ is not matched with ‘twocolumn’ . . . . .	802
1994-01-25 ltfssbas.dtx v2.1b			
<code>\math@version</code> : Corrections for math setup . . . . .	412	<code>\DeclareFontEncoding</code> : revert catcode settings earlier . . . . .	405
1994-01-25 ltmath.dtx LaTeX2e			
<code>\bordermatrix</code> : Removed <code>\p@renwd</code> . . . . .	633	<code>\DeclareFontShape@</code> : revert catcode settings earlier . . . . .	402
1994-01-26 lfsstrc.dtx v2.3c			
<code>\check@mathfonts</code> : Correct trace info placement . . . . .	459	<code>\@makespecialcolbox</code> : boxmaxdepth setting added . . . . .	911
		boxmaxdepth setting removed . . . . .	910
		General: Documentation and tasks tidied. . . . .	885

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. . . . .	789	\@mpfootnotetext: Extra group for color . . . . .	672
\@fileswithoptions: Renamed \compatibility to \@documentclasshook. ASAJ. . .	808	1994-03-07 ltboxes.dtx v1.0a	
1994-02-10 ltfsbas.dtx v2.1d		General: Unify format with other Kernel files . . . . .	661
\addto@hook: Made \addto@hook long. . . . .	426	1994-03-07 ltdefns.dtx v1.0a	
1994-02-10 ltfsccmp.dtx v2.1d		\@italiccorr: Macro added . . . . .	74
\scan@fontshape: scan away stuff after pt . . . . .	475	1994-03-07 ltfssini.dtx v2.1g	
1994-02-22 ltfssini.dtx v2.1g		General: Initial version, split from latex.dtx . . . . .	324
General: Correct error message . . .	534	Long lines wrapped to 72 columns	324
1994-02-24 ltfsbas.dtx v2.1e		1994-03-07 ltfinal.dtx v0.1a	
\DeclareFontShape: Separate restoration of catcodes for fd cmds . . . . .	402	General: Add code from the old dump.dtx . . . . .	975
\define@newfont: Separate restoration of catcodes for fd cmds . . . . .	415	Initial version, split from latex.dtx	961
\@fss@catcodes: Separate restoration of catcodes for fd cmds . . . . .	415	move code here from lhyphen.dtx	967
1994-02-25 ltdirchk.dtx v0.2j		Remove oldcomments environment	961
General: Remove need for drv file . . .	1	use \InputIfFileExists not \IfFileExists . . . . .	967
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 . . . . .	759
1994-03-02 ltboxes.dtx v0.1e		\@footnotetext: (DPC) Extra group for colour . . . . .	769
General: Add 2ekernel module . . . . .	661	\@xfloat: (DPC) Extra group for colour . . . . .	755
Remove need for drv file . . . . .	661	1994-03-07 lthyphen.dtx v0.1c	
1994-03-02 ltclass.dtx v0.3a		General: move the 2ekernel code to ltfinal.dtx . . . . .	959
General: Remove need for driver file . . .	789	1994-03-07 ltlength.dtx v1.0a	
1994-03-03 ltboxes.dtx v0.1f		\@settdim: (DPC) Extra group for colour . . . . .	399
\@irsbox: Replaced a missing \else . . .	674	1994-03-07 ltlists.dtx v1.0a	
1994-03-04 ltfloat.dtx v1.0a		General: Initial version, split from latex.dtx . . . . .	645
General: Initial version, split from latex.dtx . . . . .	750	Long lines wrapped to 72 columns	645
1994-03-04 ltsect.dtx v1.0a		1994-03-07 ltpage.dtx v1.0a	
General: Initial version, split from latex.dtx . . . . .	737	General: Initial version, split from ltherest.dtx . . . . .	781
1994-03-04 lttab.dtx v1.0a		1994-03-07 ltpictur.dtx v0.1a	
General: Initial version, split from latex.dtx . . . . .	676	General: Initial version, split from latex.dtx . . . . .	701
1994-03-04 ltvers.dtx v1.0a		Long lines wrapped to 72 columns	701
General: Initial version, split from latex.dtx . . . . .	36	1994-03-07 ltsect.dtx v1.0a	
		\@hangfrom: (DPC)Extra groups for colour . . . . .	744
		1994-03-07 lttab.dtx v1.0a	
		General: Long lines wrapped to 72 columns . . . . .	676
		1994-03-08 ltclass.dtx v0.3b	
		General: Modify driver code into ‘new style’ . . . . .	789

1994-03-08 ltdirchk.dtx v1.0a	\listfiles: Reset \addtofilelist at begin document . . . . .	344
General: Reorganize driver module into ‘new style’ . . . . .	1	
1994-03-08 lplain.dtx v1.0a	General: Remove need for a driver file.	14
1994-03-10 ltfsbas.dtx v2.2f		
\math@egroup: Changed \begingroup\endgroup to \bgroup\egroup. . . . .	424	
1994-03-11 ltfsdcl.dtx v2.1b		
\DeclareSymbolFontAlphabet@: Added check against use of alphabet switch outside of math mode. . . . .	507	
\SetMathAlphabet@: Changed parameter template in temporary macro to catch check add below.	496	
1994-03-12 ltclass.dtx v0.3c		
General: Change name from docclass to ltclass . . . . .	789	
\ProvidesFile: Add \wlog . . . . .	799	
\ProvidesPackage: Add \wlog . . . . .	797	
use \gtempa . . . . .	797	
1994-03-12 ldefsns.dtx v1.0b		
\@reargdef: New defn, in terms of \@yargdef . . . . .	78	
\@yargd@f: Name changed from \XXX@argdef . . . . .	78	
1994-03-12 ltdirchk.dtx v1.0b		
General: Change name from dircheck.dtx . . . . .	1	
Minor edits to the typeouts in ltxcheck . . . . .	1	
1994-03-12 ltfloat.dtx v1.0b		
\@savemarbox: (DPC) Extra group for colour . . . . .	763	
\@xympar: (DPC) Extra bgroup for colour . . . . .	764	
1994-03-12 lplain.dtx v1.0b		
General: Name changed from lplain. The end of an era . . . . .	14	
1994-03-12 lplain.dtx v1.0e		
General: Replaced remaining width, height, depth by L <sup>A</sup> T <sub>E</sub> X macro names to save tokens. . . . .	14	
1994-03-13 lcntrnl.dtx v1.0c		
\@tfor: (DPC) Add \otf@r so a single group is correctly treated. . . . .	274	
1994-03-13 ltfilehook.dtx v0.3b		
\unqu@tefilef@nd: Use new cmd \addtofilelist . . . . .	845	
1994-03-13 ltfiles.dtx LaTeX2e		
\@addtofilelist: Macro added . . .	343	
	\@isavebox: Use \color@setgroup .	665
	\@isavepicbox: Use \color@setgroup . . . . .	665
	\color@begingroup: macro added for color support . . . . .	664
	\color@endgroup: macro added for color support . . . . .	664
	\lrbbox: Use \color@setgroup . . . .	665
	\sbox: Use \color@setgroup . . . .	665
	1994-03-14 ltfloat.dtx 1.0c	
	\@xympar: (DPC) Use \color@begingroup . . . . .	764
	1994-03-14 ltfloat.dtx v1.0c	
	\@endifloatbox: (DPC) Use \color@endgroup . . . . .	759
	\@footnotetext: (DPC) Use \color@begingroup, add \endgraf . . . . .	769
	\@savemarbox: (DPC) Use \color@begingroup . . . . .	763
	\@xfloat: (DPC) Use \color@begingroup . . . . .	755
	1994-03-15 ltfiles.dtx LaTeX2e	
	\@missingfileerror: Quit on x or X just like a real error . . . . .	341
	1994-03-15 ltntcmd.dtx v3.2a	
	General: Adapted to mass formatting Changed \/ to \@@italiccorr . . .	561
	Removed \crenewfontswitch . . .	561
	Removed defs of short-forms and all sizes except \normalize . . . . .	561
	1994-03-15 ltoutput.dtx v1.0l	
	\@addtocurcol: Changed \addvspace to \vskip . . . . .	926, 930
	\@combinedblfloats: Removed boxmaxdepth setting. . . . .	917
	\@makecol: \maxdepth changed to \@maxdepth . . . . .	909
	Removed boxmaxdepth setting. . .	909
	\@makespecialcolbox: Removed boxmaxdepth setting. . . . .	911
	\@topnewpage: Corrected and amended warning message . . . .	901
	Warning added: it should be improved . . . . .	902

General: Added some warnings when page gets full of top floats. ....	885	1994-03-29 ltcnts.dtx v1.0c General: Create file from parts of ltmiscen and ltherest. ....	391
Driver added and further tidying.	885	1994-03-29 ltlength.dtx v1.0c General: Create file ltcntlen from parts of ltmiscen and ltherest. ....	399
Removed duplicated code and corrected docstrip options. ....	885	1994-03-29 ltmiscen.dtx v1.0d General: Remove counter macros to ltcntlen .....	608
Some boxmaxdepth settings removed. ....	885	1994-03-29 ltpageno.dtx v1.0c General: Create file ltcntlen from parts of ltmiscen and ltherest. ....	602
1994-03-16 ltclass.dtx v0.3f General: Add pkgindoc package ...	834	1994-03-29 ltxref.dtx v1.0c General: Create file ltcntlen from parts of ltmiscen and ltherest. ....	603
1994-03-16 ltfiles.dtx LaTeX2e \listfiles: Move this code directly into \document .....	344	1994-03-31 ltbibl.dtx v1.0a General: Initial version of ltidxbib.dtx, split from ltherest.dtx .....	776
1994-03-16 ltfiles.dtx v1.0c \document: (DPC) directly add file list settings .....	328	1994-03-31 ltidxglo.dtx v1.0a General: Initial version of ltidxbib.dtx, split from ltherest.dtx .....	773
1994-03-16 ltmiscen.dtx v1.0b \verbatim: Remove \global\@inlabelfalse again. .	623	1994-04-09 ltcnts.dtx v1.0d \newctr: \@nocnterr now has counter name argument .....	392
1994-03-28 ltalloc.dtx v1.0d General: Redefinition of 'new' allocations removed. ....	269	\addtocounter: \@nocnterr now has counter name argument .....	392
1994-03-28 ltdirchk.dtx v1.0d General: Improve documentation ...	1	\setcounter: \@nocnterr now has counter name argument .....	392
1994-03-28 lterror.dtx v1.0d \@invalidchar: (DPC) Comment out (use catcode15 instead) .....	285	\stepcounter: Use \addtocounter to have name checked .....	393
General: Remove test for \inputlineno undefined. ....	281	1994-04-09 lthm.dtx v1.0b \othm: Use standard counter error message (FMi) .....	735
1994-03-28 ltfiles.dtx v1.0d \document: (DPC) Use \normalsize not \@normalsize .....	327	1994-04-11 ltclass.dtx v0.3g \endfilecontents: Add star form, don't write \endinput at the end of the file. ....	818
(DPC) remove \@normalsize check .....	327	\ProvidesFile: Protect against weird catcodes. ....	799
1994-03-28 ltfloat.dtx v1.0b \caption: Use \normalsize not \@normalsize .....	753	1994-04-11 ltfssbas.dtx v2.1h General: Added \defaultscriptratio and \defaultscriptscriptratio. ASAJ. ....	401
General: Split further from ltherest.dtx .....	750	\defaultscriptratio: Macro added	424
1994-03-28 ltlists.dtx v1.0b General: Improve documentation ...	644	\defaultscriptscriptratio: Macro added .....	424
1994-03-28 ltmiscen.dtx v1.0c General: Improve Documentation ...	608	1994-04-12 ltboxes.dtx v1.0c General: Remove \@acci, now defined in ltplain.dtx .....	669
1994-03-28 ltplain.dtx v1.0c \newlanguage: Remove some \outer declarations. ....	17	Remove \@dischyp, now defined in ltinit.dtx .....	669
1994-03-28 ltsect.dtx v1.0b General: Split further from ltherest.dtx .....	737	1994-04-12 ltdefns.dtx v1.0g \dischyp: Define \@dischyp, was previously in ltboxes.dtx .....	100
1994-03-28 lttab.dtx v1.0b General: Improve documentation ...	676		
1994-03-28 ltthm.dtx v1.0a General: Initial version, split from latex.dtx .....	733		

1994-04-12 ltplain.dtx v1.0d		\fontsubfuzz: Changed dimen to macro . . . . .	468
General: Define \acci . . . . .	31	\subst@size: \font@submax and \fontsubfuzz now macros . . . . .	469
1994-04-12 ltvers.dtx v1.0b		1994-04-19 ltpage.dtx v1.0b	
General: Have version info generated automatically . . . . .	36	General: Improve documentation . . . . .	781
1994-04-14 lftntcmd.dtx v3.2b		1994-04-20 lftntcmd.dtx v3.3a	
General: Macros renamed to non-private forms, JB . . . . .	561	General: Documentation up-dated . . . . .	561
\DeclareOldFontCommand: Renamed from \newfontswitch . . . . .	568	New implementation of \nocorr . . . . .	561
1994-04-15 ltboxes.dtx v1.0d		\check@nocorr@: Macros added . . . . .	565
\@isavebox: Added missing percent character . . . . .	665	\maybe@ic@: \nocorr etc removed from list of tokens to check, leaving only punctuation characters . . . . .	566
1994-04-17 ltcnts.dtx v1.0e		1994-04-20 ltmiscen.dtx v1.0e	
\@newctr: Use \nocounterr instead of \nocnterr . . . . .	392	\enddocument@kernel@warnings: Changed logic for producing warning messages . . . . .	611
\addtocounter: Use \nocounterr instead of \nocnterr . . . . .	392	1994-04-21 ltboxes.dtx v1.0e	
\setcounter: Use \nocounterr instead of \nocnterr . . . . .	392	\@iiminipage: Extra \bgroup for color . . . . .	671
1994-04-17 lterror.dtx v1.0h		\@mpfootnotetext: Extra \endgraf for color . . . . .	672
\@nocnterr: New name for error message, old error message (without arg) kept . . . . .	282	\endminipage: Extra \egroup for color . . . . .	671
1994-04-17 ltthm.dtx v1.0c		1994-04-21 ltfinal.dtx v0.1c	
\@othm: Use new std counter error message (FMi) . . . . .	735	General: Added comments, set the catcodes of 128–255 . . . . .	961
1994-04-18 ltfinal.dtx v0.1b		1994-04-22 ltfssini.dtx v2.1g	
General: Initialise \textheight, \textwidth and page style . . . . .	964	\not@math@alphabet: Message changed again . . . . .	530
1994-04-18 ltfloat.dtx v1.0d		1994-04-23 ltfinal.dtx v0.1d	
\@footnotetext: (DPC) Remove Colour support . . . . .	769	General: Check that \font@submax is still zero . . . . .	961
\@savemarbox: (DPC) Remove Colour support . . . . .	763	1994-04-24 ltoutput.dtx v1.0m	
1994-04-18 ltfssbas.dtx v2.1i		\@resetfps: Number 2 changed to \tw@ . . . . .	948
General: Macro \no@alphabet@help removed again . . . . .	401	Warning changed . . . . .	948
\calculate@math@sizes: Changed message to log only . . . . .	424	\@specialoutput: Message changed to give more info and ‘top’ removed . . . . .	903
\no@alphabet@error: Use std LaTeX error macro . . . . .	401	\@topnewpage: Message changed to give more info . . . . .	902
1994-04-18 ltfsdcl.dtx ???		Warning message removed as it will be generated later . . . . .	901
\DeclareMathAlphabet: Pass correct arg (2 not 3) . . . . .	494	General: Changed \normalsize to \normalsize. . . . .	885
1994-04-18 ltfsdcl.dtx v2.1d		Corrected unverbed commands in documentation . . . . .	885
General: Removed surplus \no@alphabet@error (see fam.dtx) . . . . .	479	Removed some long lines and other aesthetic changes . . . . .	885
1994-04-18 lfsstrc.dtx v2.3d		Warning messages changed/corrected . . . . .	885
General: Changed to new error/warning scheme . . . . .	449	1994-04-24 ltpictur.dtx v0.1b	
\font@submax: Changed dimen to macro . . . . .	468	General: Removed surplus spaces after \hbox to in several cases . . . . .	701

1994-04-25	ltclass.dtx v0.3h		Warning changed to info message in \@protecteddef ..... 561
	General: Removed spurious extra `s at the end of error messages .....	789	
1994-04-25	ltfloat.dtx v1.0e		\@activechar@info: \@activechar@warning changed to \@activechar@info ..... 912
	\@largefloatcheck: Changed warning message to give more info .....	759	\@combinedblfloats: Removed rule in topnewpage case ..... 917
	Command added .....	759	\@emptycol: Empty column action added: \@emptycol ..... 900
	General: Changed warning messages Removed obsolete tracing code ..	750	\@fllsetnum: Rogue space removed .. 948
1994-04-27	ltfsstrc.dtx v2.3e		\@specialoutput: Cut-off point changed to 2\baselineskip .... 903
	General: Corrected item that was forgotten in last change. ....	449	Empty column action added: \@emptycol ..... 903
1994-04-28	lterror.dtx v1.0j		Extra empty column added for twocolumn case ..... 903
	\@inmatherr: Macro added .....	284	Extra empty column added for twocolumn case (wrong, see below) ..... 903
1994-04-28	lterror.dtx v1.1c		\@topnewpage: Added setting of \col@number ..... 900, 901
	\@inmatherr: Replaced \noexpand with \protect. ....	284	Cut-off point changed to 3\baselineskip ..... 902
1994-04-28	ltfssdcl.dtx v2.1e		Empty column action added: \@emptycol ..... 902
	General: Removed all \uppercase in hex num parsing macros .....	479	Message changed for Frank .....
1994-04-28	ltlists.dtx v1.0c		General: \@activechar@warning changed to an info message. .... 885
	\item: Replaced \@ltxnomath by \@inmatherr .....	655	Added \col@number. .... 885
1994-04-28	ltpictur.dtx v0.1c		Documentation tidied. .... 885
	\@multiput: (DPC) Macro added ..	705	Empty column action added. .... 885
	General: bezier curves added .....	728	Fixed bug from \dblfigrule with \@topnewpage. .... 885
	\multiput: (DPC) Ignore spaces between )( .....	704	Full of floats action improved. .... 885
	\picture: (DPC) Ignore spaces before ( .....	703	\col@number: Added \col@number .. 897
1994-04-28	lplain.dtx v1.0g		\onecolumn: Added setting of \col@number ..... 899
	General: Turn off overfull box tracing in log .....	26	1994-05-01 lterror.dtx v1.0k \@lateinfo: (CAR) Added draft \@lateinfo. .... 281
1994-04-29	ltclass.dtx v1.0a		1994-05-01 ltoutenc.dtx 1.4a General: Added the \a command. ... 357
	General: Change version number to 1 (no other change) .....	789	Added the \SaveAtCatcode and \RestoreAtCatcode commands. .. 361
1994-04-29	ltmiscen.dtx v1.0f		Removed the uc/lc table settings, since the T1 uc/lc table is now the default. .... 369
	\@verbatim: \leavevmode added ..	624	Rewrote for the new syntax. .. 361, 363
	Change to \everypar added .....	624	1994-05-01 ltoutenc.dtx v1.4a General: Removed Rokicki's encoding. .... 345
1994-04-29	ltoutenc.dtx 1.4a		Renamed the commands, removed the \EncodingSpecific command.
	General: Removed \EncodingSpecific. Renamed all the commands. Added \DeclareTextGlyph and \UndeclareTextCommand. ....	349	
	Removed Rokicki's OT1 variant encoding. Moved the driver to the top. ....	348	
1994-04-30	ltfntcmd.dtx v3.3b		
	General: Documentation up-dated and tidied .....	561	
	Prefix frag@ changed to frag in \@protecteddef .....	561	
	Title changed .....	561	

Turned all slots into decimal.	1994-05-03 ltmiscen.dtx v1.0h
Added \a. .... 345	\@centercr: \@badcrerr replaced by \@nolnerr ..... 620
1994-05-02 ltcntrl.dtx v1.0l	1994-05-03 ltab.dtx v1.0d
\@break@tfor: Macro added (from ltfiles.dtx) ..... 274	\@endpbox: Use \@finalstrut based on depth of \@arstrutbox ..... 699
1994-05-02 ltdefns.dtx v1.1f	1994-05-04 ltclass.dtx v1.0b
\renewcommand: Removed surplus \space in error ..... 79	\NeedsTeXFormat: Changed wording of the warning ..... 807
\renewenvironment: Removed surplus \space in error ..... 80	1994-05-04 lterror.dtx v1.0m
1994-05-02 ltfiles.dtx v1.0f	\@badcrerr: Error message removed ..... 284
\@iffilenonpath: \@break@loop renamed to \@break@tfor ..... 339	1994-05-05 ltbibl.dtx v1.0c
\@obsoletefile: Make \@onlypreamble ..... 343	\@citex: Set switch for warning and end of run. ..... 777
1994-05-02 ltfinal.dtx v0.1e	\@nocite: Do not write page number in \@nocite warning message. ..... 778
General: Added setting the ‘letter’ catcodes. ..... 973	Set switch for warning and end of run. ..... 778
Added setting the ‘other’ catcodes. 973	1994-05-05 ltfinal.dtx v0.1g
Added setting the special catcodes. 973	General: Added empty errhelp. .... 961
Made slot 127 illegal ..... 973	\@errhelp: Set error help empty. .... 976
Set all the catcodes ..... 961	1994-05-05 lfntcmd.dtx v3.3c
1994-05-02 ltfinal.dtx v0.1f	\@math@egroup: Corrected \@fontswitch and added saved versions ..... 568
General: Set the catcode of control-J. 973	General: Corrected \@fontswitch .. 561
1994-05-02 ltmiscen.dtx v1.0g	1994-05-05 ltmiscen.dtx v1.0i
General: Changed 91 to 1991 and moved some bits ..... 608	General: Removed braces from ifnextchar and ifstar arguments .. 608
1994-05-02 ltoutput.dtx v1.0o	1994-05-07 ltab.dtx v1.0c
\@resetfps: Code shortened ..... 948	\@maxtab: Changed \@firsttab to \chardef ..... 680
General: Code of \@resetfps shortened. ..... 885	Changed \@maxtab to \chardef .. 680
1994-05-03 ltbibl.dtx v1.0b	General: Removed definition of \+ .. 676
\@nocite: Make \@nocite issue a warning for an undefined citation key. ..... 778	Removed surplus braces from \@ifnextchar constructs ..... 676
1994-05-03 ltfinal.dtx v0.1f	1994-05-08 lfntcmd.dtx v3.3d
General: Set the catcode of control-J to be ‘other’, for use in messages. 961	General: Removed \@undefinedfonterror ..... 561
1994-05-03 ltfloat.dtx v1.0f	\@normalsize: Removed \@undefinedfonterror ..... 569
General: (CAR) Added \@largefloatcheck ..... 750	1994-05-09 lfntcmd.dtx v3.3f
Removed unnecessary braces from arguments of \@ifnextchar .... 750	General: Replaced all \next by \@let@token and undo change 3.3e, whatever that was. .... 561
\@end@dblfloat: \@largefloatcheck added ..... 758	1994-05-10 ltdefns.dtx v1.0n
\@end@float: (CAR) Added \@largefloatcheck ..... 757	General: (ASAJ) Added \DeclareProtectedCommand. .... 73
1994-05-03 ltfsdcl.dtx v2.1f	Added \DeclareProtectedCommand .. 82
General: Renamed \@DeclarerMathDelimiter to \@DeclarerMathDelimiter ..... 479	Removed braces around \@ifundefined argument. ASAJ. 79
1994-05-03 ltlists.dtx v1.0d	\makeatother: Added \makeatletter and \makeatother ASAJ. .... 100
\@item: \hskip changed to \kern .. 656	
\@item: Removed superfluous braces 655	

1994-05-10 lterror.dtx v1.0n		\DeclareTextAccent: Reimplemented using \DeclareTextCommand. . . . .	351	
	\@latexerr: (ASAJ) Added extra blank lines to \@latexerr. . . . .	281		
1994-05-10 ltmiscen.dtx v1.0j		\hspace: Use \DeclareRobustCommand. ASAJ. . . . .	320	
	\@osverb: Slight change in error message text. . . . .	626		
1994-05-11 ltboxes.dtx v1.0f		\@finalstrut: macro added . . . . .	675	
	\begin@tempboxa: Use new \color@setgroup concept. . . . .	662		
	\@iiiminipage: Use new \color@setgroup concept. . . . .	671		
	\@mpfootnotetext: Use new \color@setgroup concept. . . . .	672		
	Use new \normalcolor and \@finalstrut. . . . .	672		
General: Superfluous braces removed from several commands . . . . .	661	General: (ASAJ) Fixed a bug with \relax which was using \@gobble before defining it. . . . .	73	
\color@setgroup: macro added for color support . . . . .	664	Fixed a bug with \relax which was using \@gobble before defining it. . . . .	82	
\endminipage: Use new \color@setgroup concept. . . . .	671	1994-05-12 ltfssbas.dtx v2.1j		
1994-05-11 ltclass.dtx v1.0c		General: New baselinestretch concept . . . . .	401	
	\endfilecontents: Add checks for form feed and tab . . . . .	818	Replaced hand-protected commands by \DeclareRobustCommand defs . . . . .	401
1994-05-11 ltdirchk.dtx v1.0e		\f@linespread: New macro . . . . .	411	
General: Add \ProvidesFile as used in fd files. . . . .	4	\fontencoding: Use \DeclareRobustCommand. . . . .	409	
1994-05-11 lterror.dtx v1.0o		\fontfamily: Use \DeclareRobustCommand. . . . .	410	
\@latexerr: (ASAJ) Removed one of the extra blank lines to \@latexerr. . . . .	281	\fontseries: Use \DeclareRobustCommand. . . . .	410	
1994-05-11 ltlogos.dtx v1.0o		\fontshape: Use \DeclareRobustCommand. . . . .	410	
\LaTeX: Use \DeclareProtectedCommand. ASAJ. . . . .	323	\fontsize: Redefined to use \set@fontsize . . . . .	411	
\LaTeXe: Use \DeclareProtectedCommand. ASAJ. . . . .	323	\linespread: New macro . . . . .	411	
1994-05-11 ltoutenc.dtx 1.5a		\mathversion: Use \DeclareRobustCommand. . . . .	412	
General: Made T1 and OT1 generate packages rather than def files.		1994-05-12 ltfssdcl.dtx v2.1g		
Renamed the ‘package’ module to ‘teststy’. . . . .	348	General: Allow \relax as undefined command . . . . .	479	
1994-05-11 ltoutenc.dtx v1.5a		Allow \relax’ed cmds to be declared . . . . .	479	
General: Reimplemented \DeclareTextCommand using \@changed@cmd and \DeclareProtectedCommand. . . . .	349	1994-05-12 ltfssini.dtx v2.1i		
Renamed the commands again. Made the encoding part of the command syntax. Added the \DeclareTextCommand interface.		General: Moved \fontencoding to fam.dtx . . . . .	509	
Used \DeclareProtectedCommand. . . . .	345	Moved \fontfamily to fam.dtx . . . . .	509	
		Moved \fontseries to fam.dtx . . . . .	509	
		Moved \fontshape to fam.dtx . . . . .	509	
		Moved \fontsize to fam.dtx . . . . .	509	
		Moved \mathversion to fam.dtx . . . . .	509	
		Moved \selectfont to tracefnt.dtx	509	

1994-05-12 ltfssrc.dtx v2.3f		1994-05-13 ltfinal.dtx v1.0h	
\selectfont: Use \DeclareRobustCommand . . . . .	453	General: Added output enc stuff . . . . .	975
1994-05-12 ltoutenc.dtx v1.5a		1994-05-13 ltffloat.dtx v1.0g	
General: Removed the \SaveAtCatcode and \RestoreAtCatcode commands. . . . .	361	\@footnotetext: (DPC) Add new style colour support: \normalcolor . . . . .	769
Rewrote for the new syntax. . . . .	361, 363	(DPC) Use \finalstrut . . . . .	769
1994-05-12 ltoutput.dtx v1.0p		\@xfloat: (DPC) Use \normalcolor	755
\@writesetup: \normalcolor added	912	1994-05-13 ltfntcmd.dtx v3.3g	
General: \normalcolor added in various places (DPC). . . . .	885	General: Replaced \protecteddef by \DeclareRobustCommand . . . . .	561
1994-05-13 ltboxes.dtx v1.0h		1994-05-13 lfssbas.dtx v2.1k	
\@arrayparboxrestore: New accent system, use \let not \def . . . . .	670	General: Remove File identification 'typeout' . . . . .	401
1994-05-13 ltcounds.dtx v1.0f		1994-05-13 lfssbas.dtx v2.1l	
General: Removed \Ialpha . . . . .	396	\DeclareFontEncoding: Init encoding change command . . . . .	405
Removed \ialpha . . . . .	396	\define@newfont: Use \input@ for fd files . . . . .	415
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 . . . . .	479
Removed \if@short@command. . . . .	73	1994-05-13 lfssini.dtx v2.1j	
(ASAJ) Replaces \space by ' ' in \csname. . . . .	73	General: Removed file identification typeout . . . . .	509
Renamed \DeclareProtectedCommand to \DeclareRobustCommand.		1994-05-13 ltfssrc.dtx v2.3g	
Removed \if@short@command.		General: Removed timeouts as \ProvidesPackage writes to log. . . . .	449
Moved to after the definition of \@gobble. . . . .	82	1994-05-13 ltoutenc.dtx v1.5b	
1994-05-13 ltdefns.dtx v1.0r		General: Added \{, \} and \\$. . . . .	345
General: (ASAJ) Added logging message to \DeclareProtectedCommand. . . . .	73	Renamed \DeclareProtectedCommand to \DeclareRobustCommand. . . . .	345
Added logging message to \DeclareProtectedCommand. . . . .	82	Replaces \space by ' ' in \csname. . . . .	345
1994-05-13 ltdefns.dtx v1.0s		1994-05-13 ltpictur.dtx v0.1d	
General: (ASAJ) Added \backslashchar. . . . .	73	General: Removed surplus braces from \if.. constructions . . . . .	701
(ASAJ) Coded \ifdefinable more efficiently. . . . .	73	1994-05-13 ltab.dtx v1.0d	
Coded more efficiently, thanks to FMi. . . . .	79	\@contfield: Colour support . . . . .	682
1994-05-13 ltfiles.dtx LaTeXe		\@startfield: Colour support . . . . .	682
\listfiles: Stop \listfiles being run twice . . . . .	343	\@stopfield: Colour support . . . . .	682
1994-05-13 ltfiles.dtx v1.0g		\@a: moved to ltoutenc . . . . .	680
\document: Added execution of \every@size . . . . .	327	1994-05-14 fontdef.dtx v2.1f	
1994-05-13 ltfinal.dtx v0.1h		General: Removed .def files. . . . .	539
General: Added package otlenc, and defined \acci, \accii and \acciii. . . . .	961	1994-05-14 lfssbas.dtx v2.1m	
		\enc@update: Macro added . . . . .	410
		1994-05-14 lfssbas.dtx v2.1n	
		General: Set defaults for all \f@... . . . .	411
		\DeclareErrorFont: Don't set \f@encoding . . . . .	418
		\@footnotetext: (DPC) Log if encoding is redeclared . . . . .	405
		Only init enc change cmd when new encoding . . . . .	405

1994-05-14 ltfssini.dtx v2.1k		1994-05-16 ltlogos.dtx v1.1a
General: Init error font just before checking for fontdef.cfg . . . . .	534	General: (ASAJ) Split from ltinit.dtx. 323
\reset@font: Remove surplus braces	532	\ensuremath: Use
1994-05-14 ltfsstrc.dtx v2.3h		\ DeclareRobustCommand and add extra braces in math mode . . . . .
\selectfont: Added \enc@update . . . . .	455	640
1994-05-14 ltoutenc.dtx 1.5d		1994-05-16 ltoutenc.dtx 1.5h
General: Moved the driver to the top. 348		General: \pounds was still using u rather than ui shape. . . . .
1994-05-14 ltoutenc.dtx v1.5c		361
General: Added the fontenc package . . . . .	388	1994-05-16 ltoutenc.dtx v1.5f
Added the fontenc package. . . . .	345	General: enc files now have uc encoding name parts (FMi) . . . . .
Fixed a bug which caused an infinite loop if \f@encoding was incorrectly set. . . . .	345, 349	345
Moved fontsmp to its own dtx file. 345		Revert code so that the encoding given is used in \ DeclareTextCommand (FMi) . . . . .
1994-05-14 ltoutenc.dtx v1.5d		345
General: Rewrote \ DeclareTextCommand to define its argument to use the current encoding by default, rather than the encoding provided to \ DeclareTextCommand. . . . .	349	1994-05-16 ltoutenc.dtx v1.5g
Tidied up the documentation. . . . .	345	General: Made fontenc.sty use the new mixed-case encoding files. . . . .
1994-05-14 ltoutenc.dtx v1.5e		345
General: Replaced \ENC@cmd by \ENC-cmd. . . . .	345	Removed the lowercasing of the filename. . . . .
1994-05-15 ltfsbas.dtx v2.1o		388
General: encoding cmd changed to enc-cmd . . . . .	401	1994-05-16 ltoutenc.dtx v1.5h
1994-05-16 fontdef.dtx v2.1g		General: Added \NG, \ng, \TH, \th, \DH, \dh, \DJ and \dj. . . . .
General: Removed \DeclareFontEncoding for ot1 and t1 and input .def files instead . . .	539	345
1994-05-16 ltalloc.dtx v1.1a		Added \r (ring accent) and \k (ogonek) accents. . . . .
General: (ASAJ) Split from ltinit.dtx. 269		345
1994-05-16 ltcntrl.dtx v1.0a		Fixed a bug with \pounds. . . . .
General: (ASAJ) Split from ltinit.dtx. 271		345
1994-05-16 ltfdfns.dtx v1.1a		Removed \P from the OT1 definitions file. . . . .
General: (ASAJ) Split from ltinit.dtx. 73		345
1994-05-16 lterror.dtx v1.1a		1994-05-16 ltoutenc.dtx v1.5i
General: (ASAJ) Completely new error interface. . . . .	275	General: Fixed a bug with \d. . . . .
(ASAJ) Split from ltinit.dtx. . . . .	275	345
1994-05-16 ltfinal.dtx v1.0i		1994-05-16 ltoutput.dtx v1.0q
General: moved output enc stuff to lffonts . . . . .	975	\@writetop: Changed setting of accents (FMi): with the new encoding setup they can use \let. It could also use the new internal commands? . . . . .
1994-05-16 ltfsbas.dtx v2.1p		913
\fontsize: Pass \baselinestretch not \f@linespread . . . . .	411	General: Changed setting of accents (FMi). . . . .
\linespread: Remove surplus braces	411	885
1994-05-16 ltfssini.dtx v2.1m		1994-05-16 ltpar.dtx v1.1a
\@accii: Define saved versions of accents . . . . .	535	General: (ASAJ) Split from ltinit.dtx. 286
		1994-05-16 lplain.dtx v1.0h
		General: Comment out encoding specific commands . . . . .
		30
		Remove \@acci and friends again .
		31
		Remove unnecessary def for \item
		30
		\loop: Use Kabelschacht method . . .
		28
		\m@th: Remove unnecessary space . . .
		30
		1994-05-16 ltspace.dtx v1.1a
		General: (ASAJ) Split from ltinit.dtx. 304
		1994-05-17 ltclass.dtx v1.0e
		\@use@option: Execute option after removing from list, not before . . .
		804

1994-05-17 ltdefns.dtx 1.1b General: (ASAJ) Added the \@protect@... commands. . . . .	83	1994-05-19 ltcntlen.dtx v1.1a General: Extracted file from ltcntlen. . . . .	391
1994-05-17 ltdefns.dtx v1.1b General: (ASAJ) Added definitions for protect. . . . .	73	1994-05-19 ltdefns.dtx v1.1d General: (RmS) Added definitions for \@namedef and \@nameuse again. . . . .	73
(ASAJ) Removed warnings and logging to lterror.dtx. . . . .	73	1994-05-19 ltfinal.dtx v0.1k General: Removed \makeat... . . . . .	961
Added the discussion of protected commands, defined the values that \protect should have. . . . .	83	1994-05-19 ltidxglo.dtx v1.1a General: Initial version of ltidxglo.dtx, split from ltidxbib.dtx . . . . .	773
1994-05-17 ltdefns.dtx v1.1c General: (ASAJ) Redid definitions for protect. . . . .	73	1994-05-19 ltlenth.dtx v1.1a General: Extract file ltlenth from ltcntlen. . . . .	399
1994-05-17 lterror.dtx v1.1b General: (ASAJ) Moved error stuff from ltdefns.dtx. . . . .	275	1994-05-19 ltpageno.dtx v1.1a General: Extract file ltpageno from ltcntlen. . . . .	602
1994-05-17 ltfsmini.dtx v2.1n \copyright: Really add extra braces . . . . .	532	1994-05-19 lplain.dtx v0.1k ltfinal \showoutput: used \maxdimen not 99999 . . . . .	32
\nfss@text: Added braces to allow use in subscripts . . . . .	532	\showoverfull: used \one not 1 . . . . .	32
1994-05-17 ltmath.dtx v1.0i General: Replaced \let by \gdef, for indirect definition. . . . .	636	1994-05-19 ltxref.dtx v1.1a General: Extract file ltxref from ltcntlen. . . . .	603
1994-05-17 ltoutenc.dtx v1.5j General: Added braces to \pounds so it works as a subscript. . . . .	345	1994-05-20 ltdefns.dtx v1.1e General: Changed command name from \@checkcommand to \CheckCommand. . . . .	73
1994-05-18 ltdefns.dtx 1.1c General: (ASAJ) Renamed the commands, and removed one which is no longer needed. . . . .	83	\CheckCommand: Changed name from \@checkcommand to \CheckCommand. . . . .	81
1994-05-18 ltdefns.dtx v1.1c General: Redid the discussion and definitions, in line with the proposed new setting of \protect in the output routine. . . . .	83	1994-05-20 lterror.dtx v1.1c General: (ASAJ) Added \@later@info@no@line. . . . .	275
1994-05-18 ltfinal.dtx v0.1j General: Corrected the lccode for d-bar. . . . .	961	(ASAJ) Added missing full stops. . . . .	275
1994-05-18 ltlogos.dtx v1.1b General: (ASAJ) Added the L <sup>E</sup> T <sub>E</sub> X 2 <sub>E</sub> logo use	323	(ASAJ) Fixed a bug with \@inmatherr. . . . .	275
the text font '2' rather than the math font '2'. . . . .	323	1994-05-20 ltfinal.dtx v0.1l General: Use new font warning commands . . . . .	968
1994-05-18 ltoutenc.dtx v1.5k General: Made dotted-i produce 'i'. . . . .	345	1994-05-20 ltfloat.dtx v1.0h \@endfloatbox: Restore outer value of @nobreak switch. . . . .	759
Removed braces from \pounds and \\$\\$. . . . .	345	\outer@nobreak: Macro added: default is to do nothing. . . . .	759
Replaced \defaultencoding with \encodingdefault. . . . .	345	1994-05-20 lftntcmd.dtx v3.3h General: Use new error commands . . . . .	561
1994-05-19 ltbibl.dtx v1.1a General: Initial version of ltbibl.dtx, split from ltidxbib.dtx . . . . .	776	1994-05-20 lfssbas.dtx v2.1q General: Use new error commands . . . . .	401
		1994-05-20 lfsstrc.dtx v2.3i General: Use new error command names . . . . .	449
		1994-05-20 ltmiscen.dtx v1.0l \@writefile: Added correct setting of \protect. . . . .	614

1994-05-20 ltmiscen.dtx v1.0m	\GenericError, \GenericWarning and \GenericInfo. ....	275
General: Use new warning commands		608
1994-05-20 ltoutput.dtx v1.0s	(ASAJ) Replaces \string by \protect in some messages. . .	275
\@writesetup: Added setting of \protect during \shipout. ....	912	
General: Added setting of \protect during \shipout. ....	885	
1994-05-20 ltpage.dtx v1.0d		
\markright: Changed setting for \protect. ....	782	
1994-05-20 ltsect.dtx v1.0c		
General: Correct setting of \protect.	747	
\addcontentsline: Correct setting of \protect. ....	746	
1994-05-21 ltbibl.dtx v1.1b		
General: Use new warning commands	776	
1994-05-21 lterror.dtx v1.1d		
General: (ASAJ) Made the error commands robust. ....	275	
1994-05-21 ltfiles.dtx v1.0h		
General: Use new error commands	324	
1994-05-21 ltlists.dtx v1.0f		
General: Use new error commands	644	
1994-05-21 ltmiscen.dtx v1.0n		
General: Use new error commands	608	
1994-05-21 ltsect.dtx v1.0d		
General: Use new error commands	737	
1994-05-21 ltab.dtx v1.0f		
General: Use new error commands	676	
1994-05-21 ltxref.dtx v1.1b		
General: Use new warning commands	603	
\newlabel: Use new warning commands ....	605	
1994-05-22 ltclass.dtx v1.0f		
General: Use new warning and error commands ....	785	
1994-05-22 ltdefns.dtx v1.1f		
General: Use new warning and error cmds ....	73	
1994-05-22 lterror.dtx v1.1e		
General: (ASAJ) Replaced bgroup by begingroup in error messages, to stop extra mathords creeping into math mode. ....	275	
1994-05-22 lterror.dtx v1.2a		
General: (ASAJ) Made \GenericError, \GenericWarning and \GenericInfo robust. ....	275	
(ASAJ) Replaced \\ and tilde by \MessageBreak and \space. ....	275	
(ASAJ) Replaced \@generic@message and \@generic@error by		
1994-05-22 ltfloat.dtx v1.0i	\GenericError: (DPC) Alternative version added for old TeXs . .	276
1994-05-22 ltfloat.dtx v1.0t	(DPC) New version using long command name. ....	276
1994-05-22 ltpictur.dtx v0.1e		
General: Use new warning cmd ..	701	
1994-05-23 ltclass.dtx v1.0h		
\NeedsTeXFormat: Don't stop completely when format is wrong	807	
\usepackage: Remove argument if possible ....	807	
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 ....	467	
1994-05-23 ltoutput.dtx v1.0u		
\@activechar@info: Added \MessageBreak ....	912	
\@writesetup: Changed resetting of \protect after shipout to use \aftergroup ....	912	
General: Added \MessageBreak. ....	885	
Changed resetting of \protect after shipout. ....	885	
1994-05-24 lterror.dtx v1.2e		
\@latex@info@no@line: Macro added	279	
1994-05-24 lterror.dtx v1.2f		
General: (DPC) wrap long lines ..	275	
1994-05-24 ltntcmd.dtx v3.3i		
General: Tidying and typos fixed ..	561	
1994-05-24 ltmiscen.dtx v1.0q		
\@currenvline: Use \empty as outer default ....	619	
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 .....	561	
\check@icr: Macros added .....	565	

\check@nocorr@: Insertion of \aftergroups moved and defaults set up for efficiency . . . . .	565	1994-06-08 ltfinal.dtx v1.0m General: Add patch file system . . . . .	975
\DeclareTextFontCommand: \expandafter inserted . . . . .	563	1994-06-09 ltfinal.dtx v1.0n General: For T <sub>E</sub> X2, do not set codes for higher half of character table. . . . .	966, 973
1994-05-25 ltoutput.dtx v1.0v General: Extra documentation. . . . .	885	1994-06-09 ltfntcmd.dtx v3.3k General: Tidying and typos fixed in documentation . . . . .	561
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.	748	1994-06-18 ltfntcmd.dtx v3.3l General: Added check for empty text	561
1994-05-25 ltthm.dtx v1.0c General: Modify documentation . . .	733	\check@nocorr@: Added check for empty text . . . . .	565
1994-05-25 ltvers.dtx v1.0d General: Remove PRELIMINARY TEST RELEASE from startup banner (spring is here) . . . . .	36	1994-06-22 ltfntcmd.dtx v3.3m General: Removed space from \nfss@text . . . . .	561
1994-05-25 ltxref.dtx v1.1c General: Modify documentation . . .	603	Renamed \check@nocorr . . . . .	561
1994-05-26 ltfiles.dtx LaTeX2e \@missingfileerror: Modify message format . . . . .	341	\check@nocorr@: Renamed \check@nocorr to \text@command to improve \long error message .	565
1994-05-26 ltlogos.dtx v1.1c General: Remove \SLiTeX logo . . . .	323	\DeclareTextFontCommand: Removed space from \nfss@text . . . . .	563
1994-05-26 ltmiscen.dtx v1.0r General: \literal removed . . . . .	628	1994-06-22 ltmath.dtx v1.2t classes \mathindent: Set \mathindent at the end of the class instead of at begin document . . . . .	641
1994-05-26 ltplain.dtx v1.1m \iterate: (CAR) added \long . . . .	28	1994-07-20 ltlogos.dtx v1.1e \LaTeX: Save a few tokens . . . . .	323
\underbar: (CAR/FMi) changed to use box \tw@ . . . . .	30	\LaTeXe: Save a few tokens . . . . .	323
1994-05-26 ltplain.dtx v1.1p \underbar: (DPC) changed to use \sbox . . . . .	30	1994-07-20 ltpage.dtx v1.0h \sloppy: Save a few tokens . . . . .	783
1994-05-29 ltfssdcl.dtx v2.1j General: Use new error commands .	479	1994-09-16 ltfssbas.dtx v2.1s \nfss@catcodes: Reset [ and ] as well, just in case . . . . .	416
1994-05-31 ltfinal.dtx v1.0n General: Renamed lthyphen.* to lthyphen.*. . . . .	961	1994-10-07 ltoutenc.dtx v1.5l General: Moved the ogonek accent. .	345
1994-06-01 ltboxes.dtx v1.0i \@frameb@x: Macro added. . . . .	667	1994-10-11 ltdirchk.dtx v1.0h \@TeXversion: Check for TeX3.14 . .	13
\@iframebox: New version, so \width is correct in \framebox . . . . .	667	General: Modify all of ltxcheck again .	13
\fbox: New version, using \@frameb@x . . . . .	666	1994-10-12 ltsect.dtx v1.0f General: Doc. typos . . . . .	737
\framebox: New version, so \width is correct in \framebox . . . . .	667	1994-10-14 fontdef.dtx v2.2a General: New coding . . . . .	537
1994-06-01 ltlogos.dtx v1.1d \LaTeX: Add \m@th to force math size calculations . . . . .	323	1994-10-14 ltfssini.dtx v2.2a General: New coding for cfg files . .	509
1994-06-01 ltoutput.dtx v1.0w General: Tidied up typesetting. . . .	885	1994-10-14 ltmiscen.dtx v1.0s General: Move math to other file . .	608
		1994-10-14 ltplain.dtx v1.1a General: Moved code to other files. . .	14
		1994-10-15 ltfssbas.dtx v2.1t \extract@alpha@from@version: Warn if math alpha is used outside math . . . . .	423

1994-10-18 ltboxes.dtx v1.0j		1994-10-25 ltdefns.dtx v1.2b
\@frameb@x: \leavevmode added . . .	667	General: Documentation
\@iframebox: \leavevmode moved to \@frameb@x . . . . .	667	improvements . . . . .
\@parboxto: Macro added to remove misuse of \empty . . . . .	669	73
General: stuff from ltpatch done . . .	661	
\fbox: \long added . . . . .	666	
\mbox: \long added . . . . .	662	
\sbox: \long added . . . . .	665	
1994-10-18 ltclass.dtx v1.0j		1994-10-25 ltoutenc.dtx 1.6a
General: Move \listfiles to ltfiles.dtx . . . . .	785	General: Added \textdollar, \textlbrace, \textrbrace, \textsterling, \textunderline. . . . .
1994-10-18 ltdefns.dtx v1.2a		363
\@star@or@long: macro added . . . . .	76	Removed \textlbrace, \textrbrace, \textunderline to give them their proper names. . . . .
General: Add extra test for \endgraf . . .	73	363
Add star-forms for all commands . . . . .	73	
\renewenvironment: reset end command . . . . .	80	1994-10-25 ltoutenc.dtx v1.6a
1994-10-18 ltfiles.dtx v1.0i		General: Added
\listfiles: code moved here from ltclass . . . . .	343	\ProvideTextCommand, \UseTextSymbol, \UseTextAccent, \DeclareTextSymbolDefault, \DeclareTextAccentDefault, \DeclareTextCommandDefault, and
1994-10-18 ltoutenc.dtx v1.5l		\ProvideTextCommandDefault. . . . .
General: Added new definitions of \patterns and \hyphenation. . . . .	357	345
1994-10-18 ltoutenc.dtx v1.5m		Added the \Provide commands, and the default definitions. . . . .
General: Added new definitions of \patterns and \hyphenation. . . . .	345	349
1994-10-18 ltsect.dtx v1.0g		Added the defaults. . . . .
\@dottedtocline: Added \normalcolor for page number . . .	748	357
General: Added \normalcolor . . . . .	737	Added the files OT1enc.def, T1enc.def and OMSenc.def. . . . .
1994-10-19 ltfssbas.dtx v2.1t		357
\DeclareFontEncoding: Add missing \relax. . . . .	405	Added the OMS encoding. . . . .
1994-10-23 lfsstrc.dtx v23.k		369
\every@math@size: Renamed to \every@math@size . . . . .	458	1994-10-27 ltoutenc.dtx 1.6b
1994-10-23 ltmath.dtx v1.0l		General: Added \textasciicircum \textasciitilde \textbackslash \textbar \textbraceleft \textbraceright \textcompwordmark \textemdash \textdash \textexcldown \textgreater \texthyphenchar \texthyphen \textless \textquestiondown \textquotedblleft \textquotedblright \textquotedbl \textquoteleft \textquoteright \textunderscore \textvisiblespace . . . . .
\@eqnnum: Added \normalcolor since \eqno introduces a subgroup of the displayed math group . . . . .	638	363
\ensuremath: Remove extra braces: but see p 168 of Leslie's book . . .	640	Added: \textemdash \textdash \textexcldown \texthyphenchar \texthyphen \textquestiondown \textquotedblleft \textquotedblright \textquoteleft \textquoteright . . . . .
1994-10-24 ltboxes.dtx v1.0k		361
\fbox: Inner braces added (to fix latex/1061) . . . . .	666	1994-10-27 ltoutenc.dtx v1.5d
1994-10-25 fontdef.dtx v2.2c		General: Rewrote
General: Added OMSenc.def . . . . .	539	\DeclareTextSymbol to define its argument to use the current
1994-10-25 ltboxes.dtx v1.0l		
\@isavepicbox: missing percent (moved from ltpatch) . . . . .	665	

encoding by default, to fit with \DeclareTextCommand. . . . .	349	Rewrote \copyright to use \textcircled. . . . .	359
1994-10-27 ltoutenc.dtx v1.6b		1994-10-31 fontdef.dtx v2.2d	
General: Added \textbackslash. . . . .	369	General: Added OMLenc.def ... . . .	539
Added more defaults for OT1. . . . .	357	1994-10-31 fontdef.dtx v2.2e	
Removed the enc.def files . . . . .	345	General: ... and moved further down	539
Removed the files OT1enc.def, T1enc.def and OMSenc.def. . . . .	357	1994-10-31 ltfloor.dtx v1.1a	
Renamed \textlbrace to \textbraceleft and \textrbrace to \textbraceright. . . . .	369	\@dblfloat: Major changes since two-column and one-column cases merged . . . . .	753
1994-10-29 ltmath.dtx 1.0m		\@dblflset: Macro added . . . . .	753
General: ASAJ: Added \DeclareMathOperator. . . . .	629	Major changes to parameter parsing, setting of local variables, etc; two-column and one-column cases merged; space hacks moved	753
ASAJ: Tidied up documentation.	636	\@endfloatbox: (DPC/CAR) Extra box added to remove colour resetting from vmode . . . . .	759
1994-10-29 ltmath.dtx v1.0m		\@floatboxreset: Macro added . . .	757
General: ASAJ: Added \mathellipsis, \mathdollar and \mathsterling. . . . .	636	\@footnotetext: (DPC/CAR) Move colour setting to output routine .	769
ASAJ: Removed \dag, \ddag. . . . .	636	\@savemarbox: (DPC/CAR) Extra box added for colour . . . . .	763
ASAJ: Renamed \S and \P to \mathsection and \mathparagraph and made them \mathchardef. . . . .	636	\@setfps: Macro added . . . . .	754
1994-10-29 ltoutenc.dtx v1.6c		\@xdblfloat: Macros removed: \@dbflt, \@xdblfloat . . . . .	759
General: Added commands like \dots for use in text and math. . . . .	357	\@xfloat: (DPC/CAR) Extra box added to remove colour resetting from vmode . . . . .	755
Renamed \P, \S, \dag and \ddag to \textparagraph, \textsection, \textdagger and \textdaggerdbl. . . . .	345	Major changes, removing setting of local variables, space hacks etc; two-column and one-column cases merged . . . . .	754
1994-10-30 ltdefns.dtx v1.2c		Reset hook added . . . . .	755
\@onelvel@sanitize: Macro added .	99	\@xmpar: (DPC/CAR) Extra box added since needed for floats . . .	764
General: (CAR)\@onelvel@sanitize added . . . . .	73	\fps@dbl: Macro added . . . . .	754
1994-10-30 ltdefns.dtx v1.2f		1994-10-31 ltoutput.dtx v1.1a	
General: (DPC)\newwrite's moved to ltfiles . . . . .	73	\@makecol: (DPC/CAR) Colour resetting moved to here . . . . .	909
1994-10-30 ltmath.dtx v1.0n		\@topnewpage: (DPC/CAR) Extra box added to remove colour resetting from vmode . . . . .	901
General: ASAJ: Moved the new commands to ltoutenc. . . . .	636	(DPC/CAR) Use \color@begingroup for colour . . .	901
1994-10-30 ltoutenc.dtx v1.6d		(DPC/CAR) Use \normalcolor .	901
General: Added \DeclareTextCompositeCommand. . . . .	345	1994-11-02 ltoutenc.dtx v1.6d	
Added \textcircled. . . . .	345, 359, 369	General: Wrapped lines longer than 70 characters. . . . .	345
Added \t. . . . .	359	1994-11-03 ltclass.dtx v1.0k	
Added math commands. . . . .	345	General: Move \@missingfileerror to ltfiles . . . . .	789
Added OML encoding. . . . .	345, 359		
Added the OML encoding. . . . .	370		
Made \textless and \textgreater come from OML. . . . .	359		
Moved math commands here from ltmath. . . . .	361		
Removed \textregistered. . . . .	359		

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. . . . .	741
1994-11-03 ltfiles.dtx v1.0j		General: (ASAJ) Added \protected@xdef to \thanks. . . . .	737
\@missingfileerror: Move here from ltclass . . . . .	341	1994-11-04 ltsect.dtx v1.0h	
1994-11-04 ltboxes.dtx v1.0m		General: Added \protected@write to \addtocontents. ASAJ. . . . .	747
\@mpfootnotetext: Added \protected@edef. ASAJ. . . . .	672	\addcontentsline: Added \protected@write to \addcontentsline. ASAJ. . . . .	746
1994-11-04 ltdefns.dtx v1.2e		1994-11-04 ltab.dtx v1.0h	
General: Added \set@display@protect to \typeout. ASAJ. . . . .	73	\@mkpream: (ASAJ) Added \unexpandable@protect to \@mkpream. . . . .	695
Added commands for setting and restoring \protect. ASAJ. . . . .	85	\multicolumn: (ASAJ) added \set@typeset@protect. . . . .	691
Rewrote protected short commands using \x@protect. ASAJ. . . . .	83	1994-11-04 ltxref.dtx v1.1d	
1994-11-04 lterror.dtx v1.2g		\label: (ASAJ) Added \protected@edef . . . . .	605
General: Added \set@display@protect to \Generic* commands. ASAJ. . . . .	275	(ASAJ) Added \protected@write . . . . .	605
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. . . . .	331	\@mpfootnotetext: Color resetting for footnotes moved to endminipage: as for main page. . . . .	672
\protected@write: Macro added ASAJ. . . . .	331	\color@endbox: macro added for color support . . . . .	664
1994-11-04 ltfloat.dtx v1.1b		\color@hbox: macro added for color support . . . . .	664
\@footnotetext: (ASAJ) Added \protected@edef. . . . .	769	\endminipage: Color resetting for footnotes moved to here: as for main page. . . . .	671
\footnotemark: Added \protected@xdef to \footnotemark. . . . .	770	1994-11-05 ltboxes.dtx v1.0o	
1994-11-04 ltdxglo.dtx v1.1b		\@mpfootnotetext: Color groups restored here. . . . .	672
\@wrglossary: Added \protected@write to \@wrglossary. . . . .	775	1994-11-05 ltfloat.dtx v1.1c	
\@wrindex: Added \protected@write to \@wrindex. . . . .	774	\@dblflset: Add compatibility with old version of \xfloat. . . . .	753
General: Removed \if@filesw from \makeindex. . . . .	773	\@endfloatbox: Use new \color@hbox concept. . . . .	759
\makeglossary: Removed \if@filesw from \makeglossary. . . . .	774	\@footnotetext: Removed \normalcolor (again) . . . . .	769
1994-11-04 ltmiscen.dtx v1.0t		\@savemarbox: Use new \color@hbox concept. . . . .	763
\@writefile: Removed setting of \protect. ASAJ. . . . .	614	\@setfps: Add compatibility with old version of \xfloat. . . . .	754
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	754
General: Added \_. . . . .	360	Use new \color@hbox concept. . . . .	755
Added \mathunderscore. . . . .	361	\@xmpar: Use new \color@hbox concept. . . . .	764
1994-11-04 ltpage.dtx v1.0e			
\markright: Added \unexpandable@protect. ASAJ. . . . .	782		

1994-11-05 ltoutenc.dtx v1.6g		1994-11-10 ltbibl.dtx v1.1c	
General: Added setting of \@typeset@protect to \patterns and \hyphenation. ....	357	General: Fix \nocite{*} ..... \nocite: Fix \nocite{*} .....	776 778
1994-11-05 ltoutput.dtx v1.1b		1994-11-10 ltmath.dtx v1.2v classes	
\@topnewpage: Use new \color@hbox concept. ....	901	\eqnarray: Added value of \parskip to \abovedisplayskip to compensate for negative \topsep ....	643
\@writesetup: Change protect settings for new-style, protect-free aux-files. ....	912	1994-11-10 ltoutput.dtx v1.1e	
Use new \color@hbox concept. ....	912	\@writesetup: Modify \protect setting ....	912
1994-11-05 ltoutput.dtx v1.1c		1994-11-10 lplain.dtx v1.1b	
\@begindvi: Added macro ....	916	General: (CAR) added patch to \loop. \iterate: (CAR) added extra \relax	14 28
\@begindvibox: Added macro ....	897	1994-11-11 lspace.dtx v1.2a	
\@writesetup: Add new \AtBeginDvi concept ....	912	\\: (DPC) Make robust ....	308
\AtBeginDvi: Added macro ....	897	1994-11-12 lfnntcmd.dtx v3.3o	
1994-11-06 ltfsbas.dtx v2.1u		\normalsize: Added \MessageBreak	569
\cf@encoding: New macro ....	411	1994-11-12 llists.dtx v1.2b lspace	
\DeclareFixedFont: Renamed \every@size to \every@math@size. ....	403	\endtrivlist: Changed order of tests to make \noitemerror correct: end of an era. ....	653
1994-11-06 ltfsini.dtx v2.2b		1994-11-12 ltmiscen.dtx v1.0u	
\@setsizes: Use \@typeset@protect	531	center: Changed end macro to \def: safer and consistent ....	621
1994-11-06 lfsstrc.dtx v2.3k		flushleft: Changed end macro to \def: safer and consistent ....	622
\glb@currsize: New implementation	457	flushright: Changed end macro to \def: safer and consistent ....	622
\try@simples: New implementation	468	1994-11-12 lplain.dtx v1.1c	
\try@size@substitution: New implementation ....	468	General: Comment out more encoding specific commands ....	30
\tryis@simple: New implementation	468	1994-11-12 lspace.dtx v1.2b	
1994-11-07 fontdef.dtx v2.2f		\addpenalty: Corrected error message ....	316
General: (DPC) Add \DeclareMathSizes declarations	544	\addvspace: Corrected error message	315
(DPC) Updated to use \ProvidesFile ....	539	1994-11-13 lspace.dtx v1.2c	
1994-11-07 ltfiles.dtx v1.0m		\addpenalty: Recorrected error message ....	316
\document: Renamed \every@size to \every@math@size. ....	327	\addvspace: Recorrected error message ....	315
1994-11-07 lplain.dtx v1.0l		1994-11-14 ltoutput.dtx v1.1f	
\@unused: move here from ltdefns, remove duplicate \mainaux ....	23	\@begindvi: Use normal box register: why a box? ....	916
1994-11-07 preload.dtx v2.1e		\@begindvibox: Use normal box register: why a box? ....	897
General: (DPC) Updated to use \ProvidesFile ....	558	\@writesetup: Modify new \AtBeginDvi concept ....	912
1994-11-09 ltboxes.dtx v1.0p		General: Removed old definition of \@testfp. ....	885
\@finalstrut: Revert \finalstrut to 2.09 equivalent (from ltpatch) ..	675	1994-11-14 lspace.dtx v1.2d	
General: more color changes... ....	661	\\: (DPC) Macro modified ....	308
1994-11-09 ltfsbas.dtx v2.1v		1994-11-14 ltab.dtx v1.0i	
\@vpt: (DPC) macros added, from setsizes.dtx ....	426	\tabularnewline: (DPC) Macro added ....	690
(DPC) reduce save stack usage latex/1742 ....	426		

1994-11-16 fontdef.dtx v2.2h		1994-11-18 ltfsstrc.dtx v2.3m	
General: (DPC) Removed \{ and \}	539	General: \next to \reserved@f ...	449
1994-11-17 ltboxes.dtx v1.0q		\phantom: (DPC) colour support ...	631
General: \tempa to \reserved@a ...	661	(DPC) use \expandafter instead of \next ...	631
1994-11-17 ltclass.dtx v1.0l		\prime@s: (DPC) use \let@token instead of \next and	
General: \tempa to \reserved@a ...	785	\expandafter instead of \nxt ...	636
1994-11-17 ltcntrl.dtx v1.0b		\smash: (DPC) colour support ...	632
General: \tempa to \reserved@a ...	271	(DPC) use \expandafter instead of \next ...	632
1994-11-17 ltdefns.dtx v1.0g		1994-11-21 ltfloat.dtx v1.1f	
General: \tempa to \reserved@a ...	73	\endfloatbox: Added reset of minipage flag ...	759
1994-11-17 ltdirchk.dtx v1.0j		Corrected position of \outer@nobreak ...	759
General: \tempa to \reserved@a ...	1	\marginparreset: Macro added ...	763
1994-11-17 lterror.dtx v1.2h		\savemarbox: Added \setminipage etc ...	763
General: \tempa to \reserved@a ...	275	Added resetting of size and font ...	763
1994-11-17 ltfiles.dtx v1.0n		Changed to \color@vbox ...	763
General: \tempa to \reserved@a ...	324	Use \setnobreak etc ...	763
1994-11-17 ltfinal.dtx v1.0o		\setminipage: Macro added ...	757
General: \tempa to \reserved@a ...	961	\setnobreak: Macro added ...	757
1994-11-17 ltfloat.dtx v1.1e		\xfloat: Added \setminipage ...	755
General: \tempa to \reserved@a ...	750	Added resetting of size and font ...	755
1994-11-17 ltftcmd.dtx v3.3p		Changed to \color@vbox so that large floats overflow at the bottom ...	755
General: \tempa to \reserved@a ...	561	Missing percents reinserted after 4, 8: these are not numbers. ...	754
1994-11-17 ltftssbas.dtx v2.1w		Use \setnobreak ...	755
General: \tempa to \reserved@a ...	401	\xympar: Changed to \color@vbox	764
1994-11-17 ltfsdcl.dtx v2.1m		1994-11-21 ltoutput.dtx v1.1i	
General: \tempa to \reserved@a ...	479	\addtocurcol: Added \if@nobreak test before float box ...	926, 930
1994-11-17 ltfsstrc.dtx v2.3l		\specialoutput: Added \if@nobreak test ...	905
General: \tempa to \reserved@a ...	449	\topnewpage: Changed to \color@vbox ...	901
1994-11-17 ltmath.dtx v1.0o		1994-11-22 ltfsdcl.dtx v2.1o	
General: \tempa to \reserved@a ...	629	General: wrap long lines ...	479
1994-11-17 ltmiscen.dtx v1.0v		1994-11-22 ltoutenc.dtx v1.6i	
General: \tempa to \reserved@a ...	608	General: Corrected \dots so that there's no kerning in monowidth fonts. ...	345
1994-11-17 ltoutenc.dtx v1.6h		Corrected typo with \mathunderscore. ...	345
General: (DPC) \tempa to \reserved@a ...	345	Fixed empty accents. Again. ...	345
1994-11-17 ltoutput.dtx v1.1h		1994-11-24 ltdefns.dtx v1.2h	
General: \tempa to \reserved@a ...	885	\newenv: Added test for \endgraf ...	80
1994-11-17 ltpictur.dtx v1.0f			
General: \tempa to \reserved@a ...	701		
1994-11-17 ltsect.dtx v1.0i			
General: \tempa to \reserved@a ...	737		
1994-11-17 ltab.dtx v1.0j			
General: \tempa to \reserved@a ...	676		
1994-11-18 ltboxes.dtx v1.0r			
\color@vbox: macro added for color support ...	664		
1994-11-18 ltfinal.dtx v1.0n			
General: re-allow slots 127–255 ...	973		
1994-11-18 ltftssbas.dtx v2.1x			
General: (DPC) use \reserved@f not \next ...	401		
1994-11-18 ltfsdcl.dtx v2.1m			
\DeclareMathDelimiter: (DPC) \expandafter instead of \next ...	501		

1994-11-25 ltplain.dtx v1.1f General: (DPC) Comment out lots of obsolete code . . . . .	14	1994-12-01 ltfinal.dtx v1.0p General: Renamed lthyphen.* to hyphen.*. . . . .	961
1994-11-26 ltfloat.dtx v1.1b \footnote: (ASAJ) Added \protected@xdef. . . . .	769	1994-12-01 lthyphen.dtx v1.0g General: Rename lthyphen.ltx/cfg to hyphen.ltx/cfg . . . . .	959
1994-11-28 ltcntrl.dtx v1.0c General: Documentation improvements . . . . .	271	1994-12-01 ltplain.dtx v1.1g General: (DPC) More doc changes . . .	14
1994-11-30 ltfiles.dtx v1.0o \@dofilelist: Macro added . . . . .	344	1994-12-02 fontdef.dtx v2.2i General: Commented out \ldots.	
\listfiles: Use \@dofilelist . . . . .	343	ASAJ. . . . .	537
\nofiles: There is no \gobblethree. . . . .	331	1994-12-02 ltfssini.dtx v2.2c \copyright: \copyright is now in ltoutenc. ASAJ . . . . .	532
1994-11-30 ltfsbas.dtx v2.1y \fontshape: Use \@current@cmd in \@enc@update. ASAJ. . . . .	410	1994-12-02 ltlists.dtx v1.0e \@trivlist: RmS: Added check for looping . . . . .	652
1994-11-30 ltmath.dtx 1.0q General: ASAJ: \DeclareMathOperator moved to AMSLATEX. . . . .	629	1994-12-02 ltoutenc.dtx 1.7b General: Redefined \a properly. . . .	357
1994-11-30 ltmiscen.dtx v1.0w \enddocument@kernel@warnings: (DPC) Do warnings even for \nofiles . . . . .	610	1994-12-02 ltoutenc.dtx v1.7b General: Fixed a bug with \a. . . . .	345
\enddocument: (DPC) Use \@dofilelist . . . . .	610	1994-12-04 lthyphen.dtx v1.0h General: Documentation edits for /1989 . . . . .	959
1994-11-30 ltoutenc.dtx 1.7a General: Redefined \a for the new scheme. . . . .	357	1994-12-05 ltoutenc.dtx v1.7c General: Added braces to \textcircled. . . . .	345
1994-11-30 ltoutenc.dtx v1.6g General: Removed new definitions of \patterns and \hyphenation, since encoding-specific commands now expand in the mouth. . . . .	357	1994-12-06 ltfssbas.dtx v2.1z \DeclareFontEncoding: use \nfss@catcodes . . . . .	405
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. . . . .	345	\nfss@catcodes: Added tab char as well . . . . .	415
Always load the enc.def file, so that the default encoding for the commands will change. . . . .	388	1994-12-08 ltoutenc.dtx v1.7d General: Added \null and \sh@ft to \b and \d. . . . .	345
Redefined \@changed@cmd to expand in the mouth. . . . .	349	1994-12-08 ltab.dtx v1.0k \array: Add \tabularnewline . . . .	689
Removed \@changed@x@mouth since \@changed@x now expands in the mouth. . . . .	349	\tabularnewline: (DPC) Made it \relax . . . . .	690
Rewrote \@text@composite so it allows an empty argument, or an argument containing lots of commands. . . . .	351	1994-12-09 ltbibl.dtx v1.1d \bibliographystyle: (DPC) Allow use in preamble. . . . .	778
		1994-12-10 ltfloat.dtx v1.1g \dblfloat: Old version reinstated temporarily . . . . .	778
		\dblflset: Macro removed temporarily . . . . .	753
		Old version reinstated temporarily	753
		\setfps: Macro removed temporarily . . . . .	754
		\dblfloat: Macros reinserted temporarily . . . . .	759
		\xfloat: Old version reinstated temporarily . . . . .	754

Sanitization added temporarily . . .	754	\endfilecontents: Close input check stream: latex/1487 . . . . .	818
General: Some temps reinserted temporarily . . . . .	750	1995-04-21 ltfinal.dtx v1.0q	
\fps@dbl: Macro removed temporarily . . . . .	754	General: Allow initial patch level 0 . . . . .	976
1994-12-10 ltntcmd.dtx v3.3q		1995-04-21 ltoutenc.dtx v1.7h	
\@math@egroup: Don't read arguments . . . . .	568	General: Added \null \k latex/1274 . . . . .	345
\check@nocorr@: Use \space command for comparison . . . . .	565	1995-04-22 ltfiles.dtx v1.0p	
1994-12-10 ltssdcl.dtx v2.1p		\includeonly: Allow blanks in argument . . . . .	332
\document@select@group: Surround with braces (add fourth arg) . . . . .	485	1995-04-22 ltmiscen.dtx v1.0x	
\select@group: Surround with braces (add fourth arg) . . . . .	482	General: Removed extra def of \gobble . . . . .	608
1994-12-10 ltoutenc.dtx v1.7e		1995-04-23 ltsect.dtx v1.0j	
General: Added documentation for the OML encoding. . . . .	345	\addcontentsline: Use \contentsline internally. . . . .	746
Replaced width with \width and ditto height in vrules. . . . .	345	1995-04-24 ltbibl.dtx v1.1e	
1994-12-14 ltoutenc.dtx v1.7f		\@citex: Add \mbox to undefined case: latex/1239. . . . .	777
General: Added braces to \copyright so it works unbraced in subscripts. . . . .	345	1995-04-24 ltbibl.dtx v1.1f	
Added check for math mode in \changed@cmd. . . . .	345	\bibcite: Make \onlypreamble /1388. . . . .	777
Commented out \textasciicircum, \textasciitilde, \textbackslash, \textbar, \textgreater, \texthyphenchar, \texthyphen and \textless to save memory. . . . .	345	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) . . . . .	274
\eqnnum: Added \normalcolor . . . . .	641	1995-04-24 ltoutput.dtx v1.1j	
1995-03-03 ltoutenc.dtx 1.7g		\flo@tracemessage: Do not add to kernel unless 'trace' specified . . . . .	943
General: Corrected an error in documentation referring to the tabular rather than the tabbing environment. . . . .	357	1995-04-24 ltoutput.dtx v1.1l	
1995-04-02 ltntcmd.dtx v3.3r		\begin{vbox}: Add \vbox latex/1392 . . . . .	897
\@math@egroup: Read them again to be able to add \relax. . . . .	568	\@writeshop: Reset \\ latex/1451 (DPC) . . . . .	913
1995-04-02 ltssdcl.dtx v2.1q		1995-04-24 ltpage.dtx v1.0f	
\document@select@group: fix problem for pr/1275 . . . . .	485	\fussy: reset \emergencystretch latex/1344 . . . . .	784
\select@group: fix problem for pr/1275 . . . . .	482	1995-04-24 ltplain.dtx v1.1h	
\set@mathdelimiter: fix pr/1329 . . . . .	503	\newlanguage: Remove remaining \outer declarations. . . . .	17
1995-04-02 ltfsini.dtx v2.2d		1995-04-24 ltxref.dtx v1.1e	
\not@math@alphabet: add \noexpand to second part of message . . . . .	530	\newlabel: Make \onlypreamble for /1388. . . . .	605
1995-04-21 ltclass.dtx v1.0m		1995-04-25 ltdefns.dtx v1.2i	
\DeclareOption*: Made long /1498 . . . . .	800	\check@c: Make \long for latex/1346 . . . . .	81
		\newenvironment: Parse arguments slowly but safely /1507 . . . . .	79
		1995-04-25 ltfiles.dtx v1.0q	
		\document: Removed execution of \every@size latex/1407 . . . . .	327
		1995-04-25 ltsect.dtx v1.0k	
		\dottedtocline: Added \hbox around dots. . . . .	748
		1995-04-27 ltboxes.dtx v1.0s	
		\frameb@x: Move \leavevmode for graphics/1512 . . . . .	667

\@ifframebox: Move \leavevmode for graphics/1512 . . . . .	667	1995-05-07 ltsect.dtx v1.0o General: Use \hb@xt@ . . . . .	737
\@iirsbox: Move \leavevmode for graphics/1512 . . . . .	674	1995-05-07 ltab.dtx v1.0l General: Use \hb@xt@ . . . . .	676
\@irsbox: Move \leavevmode for graphics/1512 . . . . .	674	1995-05-08 ltbibl.dtx v1.1g \@citet: Use \@firstofone . . . . .	777
\fbox: Move \leavevmode for graphics/1512 . . . . .	666	\bibitem: Removed unnecessary braces . . . . .	777
\raisebox: Move \leavevmode for graphics/1512 . . . . .	674	\nocite: Use \@firstofone . . . . .	778
1995-04-27 ltfiles.dtx v1.0r \document: Added \global to support groups in hook . . . . .	328	1995-05-08 ltdefns.dtx v1.2k \typein: Use \@firstofone . . . . .	75
1995-04-27 ltmiscen.dtx v1.0y \enddocument: \@checkend moved after hook . . . . .	609	1995-05-08 ltdefns.dtx v1.2l \typein: Remove unnecessary braces Replace \def by \let . . . . .	75
1995-04-27 ltplain.dtx v1.1i General: Move \hang and \textindent to latex209.def . . . . .	30	1995-05-08 lfsstrc.dtx v2.3n \ifnot@nil: Use \@firstofone . . . . .	462
1995-04-29 ltcntrl.dtx v1.0e General: Moved init of \protect to ltdefns.dtx . . . . .	274	1995-05-11 fontdef.dtx v2.2j General: Updates to some plain macros . . . . .	537
Removed unused defs for \@setprotect and \@resetprotect . . . . .	274	1995-05-12 ltclass.dtx v1.0n \DeclareOption*: Use \toks@ to remove need to double hash /1557	800
1995-04-29 ltdefns.dtx v1.2j \protect: Init \protect here . . . . .	85	1995-05-12 ltfloat.dtx v1.1h \footnotemark: Add \nobreak to allow hyphenation. latex/1605 . . . . .	771
1995-04-29 ltpar.dtx v1.1b General: (TO) Comments clean-up. . . . .	286	1995-05-12 ltpictur.dtx v1.0h \pictur@: Macro added for latex/1355 . . . . .	703
1995-05-02 ltsect.dtx v1.0l \dottedtocline: Don't reset to \rmfamily . . . . .	748	1995-05-12 ltvers.dtx v1.0e General: Add autoload docstrip guards . . . . .	36
1995-05-03 ltsect.dtx v1.0m General: TO: Promoted documentation to doc.sty standard . . . . .	737	Check for format older than 1 year . . . . .	36
1995-05-06 ltsect.dtx 1.0n \secCntformat: Use \quad instead of \hskip . . . . .	743	1995-05-13 lfsstrc.dtx v2.3o General: Use single hash mark in \DeclareOption . . . . .	450
\@sect: Added \relax after \secCntformat just in case . . . . .	741	1995-05-16 ltffloat.dtx v1.1i \makefnmark: Now use \textsuperscript . . . . .	768
1995-05-07 ltboxes.dtx v1.0t General: Use \hb@xt@ . . . . .	661	\textsuperscript: Command added./pr1503 . . . . .	768
1995-05-07 ltdefns.dtx v1.2k \hb@xt@: Macro added . . . . .	74	\thefootnote: Streamlined parts of code. . . . .	767
1995-05-07 ltmath.dtx v1.0r General: Use \hb@xt@ . . . . .	629	1995-05-17 ltboxes.dtx v1.0u \irsbox: Removed surplus braces . . . . .	674
1995-05-07 ltoutput.dtx v1.1m General: Use \hb@xt@ . . . . .	885	1995-05-17 ltdefns.dtx v1.0o \g@addto@macro: Make long for latex/1522 . . . . .	102
1995-05-07 ltpictur.dtx v1.0g General: Use \hb@xt@ . . . . .	701	1995-05-17 ltlists.dtx v1.0g \item: Removed surplus braces . . . . .	656
1995-05-07 ltplain.dtx v1.1j General: Use \hb@xt@ . . . . .	14	\nbitem: Removed surplus braces . . . . .	657
		enumerate: Use \thr@@ and remove surplus braces . . . . .	658
		itemize: Use \thr@@ . . . . .	659
		\makefnmark: Added \normalfont. . . . .	768

\thempfootnote: Added \itshape . . . . .	767	\textsuperscript: Use \@textsuperscript . . . . .	768
1995-05-19 ltpictur.dtx v1.1a		1995-05-24 ltfssbas.dtx v3.0a	
General: Support autoloading feature	701	General: (DPC) Make file from previous file, fam.dtx 1995/05/20	
1995-05-20 ltcnts.dtx v1.1b		v2.2d . . . . .	401
\@definecounter: Streamlined code	393	\mathgroup: (DPC) No need to redefine \newfam as not outer . . .	401
\fnsymbol: Allowing both text and math . . . . .	397	1995-05-24 ltfsscmp.dtx v3.0a	
\fnsymbol: Streamlined code . . . . .	396	General: (DPC) Make file from previous file, fam.dtx 1995/05/20	
1995-05-20 ltcnts.dtx v1.1c		v2.2d . . . . .	474
\@definecounter: And do it right . . . . .	393	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 . . . . .	768	v2.1t . . . . .	479
Moved \normalfont to \textsuperscript . . . . .	768	1995-05-24 ltfssini.dtx v3.0a	
\textsuperscript: Use \normalfont . . . . .	768	General: (DPC) Make file from previous file, lfonts.dtx 1995/05/23	
1995-05-21 ltfssdcl.dtx v2.1t		v2.2e . . . . .	509
\DeclareMathRadical: Allow for undefined cs names . . . . .	504	\cal: (DPC) Remove definition . . . . .	535
1995-05-21 ltlists.dtx v1.0f		\mit: (DPC) Remove definition . . . . .	535
General: Moved to doc.sty standard	644	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 \sqrtsign . . . . .	638	v2.3o . . . . .	449
General: Remove \mathhexbox from this file . . . . .	634	1995-05-24 ltfssrc.dtx v3.0b	
Update some plain macros . . . . .	629	General: (DPC) Fix \ProvidesFile usage . . . . .	449
\lefteqn: Use \rlap . . . . .	640	1995-05-25 ltclass.dtx v1.0p	
\rat: Use \sqrtsign instead of \sqrt . . . . .	631	\endfilecontents: Delete \filecontents after preamble . . . . .	818
1995-05-21 ltoutenc.dtx v1.7h		1995-05-25 ltfilehook.dtx v1.0t	
\inmathwarn: Added several \onlypreamble . . . . .	349	\unquote@efilef@und: (CAR) added \long . . . . .	845
1995-05-21 ltoutenc.dtx v1.7j		1995-05-25 ltfiles.dtx v1.0s	
General: Updated some plain macros	361	\document: Added check for \topskip zero . . . . .	328
1995-05-21 lplain.dtx v1.1j		1995-05-25 ltfiles.dtx v1.0t	
General: Moved some code to other files . . . . .	14	\iffileonpath: (CAR) added \long	339
1995-05-22 lplain.dtx v1.1k		\document: Corrected typo . . . . .	328
General: Definitions of \footins and \footnoterule moved to ltfloat . . . . .	31	\IfFileExists@: (CAR) added \long	338
1995-05-22 ltab.dtx v1.1a		\nofiles: (CAR) added \long . . . . .	331
General: Support autoloading feature	676	\protected@write: (CAR) added \long . . . . .	331
1995-05-23 ltfssini.dtx v2.2e		1995-05-25 ltffloat.dtx v1.1m	
\newfont: Font assignment made local again . . . . .	531	\savebox@: (CAR) Reset settings moved to hook . . . . .	763
1995-05-24 ltdefns.dtx v1.1l		\@xfloat: (CAR) Reset settings moved to hook . . . . .	755
\newif: (DPC) New implementation . . . . .	80	1995-05-25 ltlists.dtx v1.0i	
1995-05-24 ltdefns.dtx v1.2m		\endtrivlist: Macros moved from ltspace.dtx . . . . .	653
\typein: (DPC) New implementation . . . . .	75		
1995-05-24 ltfloat.dtx v1.1l			
\@textsuperscript: Command added . . . . .	768		
General: Moved definition of \footins and \footnoterule from lplain . . . . .	767		

1995-05-25	ltmath.dtx	v1.3c classes	
\@eqnnum: replace			
\reset@font\rmfamily with			
\normalfont (PR 1578) . . . . .			
1995-05-25	ltspace.dtx	v1.2f	641
\@vbsphack: (CAR) not used so			
'removed' . . . . .			
1995-05-25	ltspc.dtx	v1.2f	314
\@vspace: (CAR) \@restorespace			
added to avoid possible infinite tail			
recursion caused by a typo in the			
argument . . . . .			
1995-05-25	ltlists.dtx	v1.2f	317
(CAR) macros modified to be more			
efficient . . . . .			
General:	Macros moved to ltlists.dtx		304
1995-05-26	ltdefns.dtx	v1.2n	
\@gobblefour: (CAR) Added \longs			
1995-05-26	ltmath.dtx	v1.0s	82
\@eqnnum: Removed \rmfamily (PR			
1578), replaced \reset@font with			
\normalfont . . . . .			
1995-05-26	ltpage.dtx	v1.0g	638
\ps@plain: removed \rmfamily (PR			
1578) . . . . .			
1995-05-27	ltfssbas.dtx	v3.0b	782
\mathgroup: (FMi) But a need to			
define \new@mathgroup . . . . .			
1995-06-05	fontdef.dtx	v2.2k	401
General:	Moved math commands from		
ltoutenc.dtx. . . . .			
1995-06-05	ltfinal.dtx	v1.0r	555
General:	Added \MakeUppercase and		
\MakeLowercase. . . . .			
1995-06-05	ltoutenc.dtx	v1.7k	961
\@inmathwarn: Removed			
\protected@cmd and replaced with			
explicit \noexpand. . . . .			
General:	Allowed		
\ProvideTextCommandDefault			
after the preamble. . . . .			
1995-06-05	ltoutenc.dtx	v1.7k	349
Commented out \textless and			
\textgreater. . . . .			
Moved math commands to			
fontdef.dtx. . . . .			
1995-06-06	ltfinal.dtx	v1.0s	361
Save some tokens in			
\textvisible{space} and			
\textunderline{score}. . . . .			
1995-06-06	ltfinal.dtx	v1.0s	359
General:	Made \MakeUppercase and		
\MakeLowercase brace their			
argument. . . . .			
1995-06-09	ltoutenc.dtx	v1.7l	961
\DeclareTextComposite: Rewrote			
\DeclareTextComposite to define			
the composite as a no-argument			
command rather than a			
two-argument command. . . . .			
1995-06-11	ltspace.dtx	v1.2g	352
\restorecr: (CAR) \relax added to			
stop silent eating of *. . . . .			
1995-06-13	ltfinal.dtx	v1.0t	322
General:	Add patch level string more		
carefully . . . . .			
Call \errorstopmode . . . . .			
1995-06-13	lt pictur.dtx	v1.1b	978
General:	Use \ProvidesFile in		
autoload . . . . .			
1995-06-14	lttab.dtx	v1.1b	701
General:	Use \ProvidesFile in		
autoload . . . . .			
1995-06-15	ltfssbas.dtx	v3.0c	676
General:	(DPC) minor documentation		
changes . . . . .			
1995-06-15	ltfsscmp.dtx	v3.0b	401
General:	(DPC) minor documentation		
edits . . . . .			
1995-06-15	ltfssdcl.dtx	v3.0b	474
General:	(DPC) minor documentation		
changes . . . . .			
1995-06-19	ltbibl.dtx	v1.1h	479
\bincite: Call \newlabel so			
repeated keys produce better			
warning. . . . .			
1995-06-19	ltclass.dtx	v1.0q	777
\documentclass: Don't redefine			
\usepackage in compat mode for			
/1634 . . . . .			
1995-06-19	ltxref.dtx	v1.1e	805
\newlabel: Use \newlabel to share			
code with \bincite . . . . .			
1995-06-28	ltfssini.dtx	v3.0b	605
General:	(DPC) Fix documentation		
typos . . . . .			
1995-06-28	ltmath.dtx	v1.0t	509
General:	minor doc edits . . . . .		
1995-07-02	ltplain.dtx	v1.1n	629
General:	Removed surplus 'by' and '='		
in various places . . . . .			
\offinterlineskip:	Replaced 1000 by		
\@m . . . . .			
\showoutput:	Use \showoverfull to		
save space . . . . .			
\tracingall:	Use \showoutput to		
save space . . . . .			
1995-07-03	ltdefns.dtx	v1.2o	32
\set@typeset@protect: Use			
\@typeset@protect for init . . . . .			

1995-07-03 ltfntcmd.dtx v3.3s		1995-07-14 ltbibl.dtx v1.1i
\text@st@ic: Use clean interface for jump ..... 567		\bibcite: Remove \onlypreamble so still defined in new \enddocument 777
1995-07-05 ltfntcmd.dtx v3.3s		1995-07-14 ltxref.dtx v1.1g
\text@st@ic: Renamed from \test@next 567		\newlabel: Remove \onlypreamble so still defined in new \enddocument ..... 605
1995-07-05 ltspace.dtx v1.2h		1995-07-19 ltfssini.dtx v3.0d
\newline: Use \break ..... 310		General: (DPC) TeX2 support .... 535
\@no@pgbk: Macro replaces \pgbk and \@nopgbk ..... 308		1995-07-20 ltboxes.dtx v1.0v
\nopagebreak: Reimplemented both using \@no@pgbk ..... 307		\@isavebox: Use \sbox ..... 665
1995-07-09 ltcntrl.dtx v1.0f		\@isavepicbox: Use \sbox ..... 665
\iforloop: Reimplemented using Kabelschacht method ..... 274		1995-07-21 ltoutput.dtx v1.1o
\iwhiledim: Reimplemented using Kabelschacht method ..... 272		\writesetup: Command added ... 912
\iwhilenum: Reimplemented using Kabelschacht method ..... 272		New, experimental, versions: need in-lining ..... 912
\iwhilesw: Reimplemented using Kabelschacht method ..... 272		1995-08-09 ltmath.dtx v1.0u
\itfor: Reimplemented using Kabelschacht method ..... 274		General: Added code for class options leqno and fleqn ..... 641
1995-07-09 ltlists.dtx v1.0j		1995-08-11 ltlength.dtx v1.1b
enumerate: Use \expandafter .... 658		General: Doc typos fixed for latex/753 ..... 399
itemize: Use \expandafter .... 659		1995-08-16 ltcntrl.dtx v1.0g
1995-07-12 ltpictur.dtx v1.1d		\break@tfor: Made long ..... 274
General: allow 2e commands in 209 mode. latex/1737 ..... 701		\forloop: Made defs long ..... 274
1995-07-13 ltdefns.dtx v1.0p		\fornoop: Made defs long ..... 274
General: Updates to documentation .. 73		\iforloop: Made defs long ..... 274
1995-07-13 ltfiles.dtx v1.0u		\iwhiledim: Made defs long ..... 272
General: Updates to docu ..... 324		Removed \@whilenoop ..... 272
1995-07-13 ltfssbas.dtx v3.0d		\iwhilenum: Made defs long ..... 272
\@defaultsubs: macro added .... 421		Removed \@whilenoop ..... 272
\@defaultsubs: macro added .... 421		\iwhilesw: Removed \@whileswnoop ..... 272
General: minor documentation changes ..... 401		\itfor: Made defs long ..... 274
\wrong@fontshape: Change a macro not a switch to flag default font substitutions ..... 420		1995-08-16 ltfiles.dtx v1.0v
1995-07-13 ltmiscen.dtx v1.0z		\document: set \maxdepth ..... 328
\centercr: Use \nobreak ..... 620		set \do globally ..... 328
\enddocument@kernel@warnings: Use \@defaultsubs instead of switch 611		set \topskip globally ..... 328
\writefile: Added missing percent and use \relax in the THEN case 614		1995-08-24 ltfssbas.dtx v3.0f
\xobeysp: Use \nobreak ..... 623		General: Added autoload code .... 401
General: Improve Documentation .. 608		1995-08-24 lfsstrc.dtx v3.0c
\enddocument: Set \setckpt to \gobbletwo instead of defining it by hand ..... 609		General: Macro \gobble@font@spec removed ..... 463
Shorten redefinition of \bibcite and \newlabel ..... 610		\tryis@simple: ..... 470
		1995-08-25 ltoutput.dtx v1.1p
		General: Support autoloading feature (FMi). ..... 885
		1995-09-01 lterror.dtx v1.2i
		General: Add autoload support ... 275
		1995-09-01 lplain.dtx v1.1m
		\empty: Use \let to save space .... 28
		\I: Use \let to save space ..... 27

1995-09-14 lplain.dtx v1.1o		1995-10-11 ltoutput.dtx v1.1r
General: Moved <code>\multispan</code> to ltab.dtx . . . . .	14	<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		898
<code>\cline</code> : (DPC) New implementation	699	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 . . . . .	535	777
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.	628	General: Clarify makebox description
1995-10-01 ltfiles.dtx LaTeXe		661
<code>\@addtofilelist</code> : Macro added . . .	343	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> . . .	98	98
<code>\@ifnextchar</code> : Use <code>\@let@token</code> . . .	97	<code>\new@command</code> : (DPC) Use <code>\@testopt</code> /1911 . . . . .
<code>\@protected@testopt</code> : Macro added .	78	77
<code>\@testopt</code> : Macro added . . . . .	77	<code>\new@environment</code> : (DPC) Use <code>\@testopt /1911</code> . . . . .
<code>\@xargdef</code> : New implementation, using <code>\@test@opt</code> . . . . .	77	79
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 . . . . .	537	75
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) . . . . .	75	<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		532
General: New autoload code . . . . .	701	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 . . . . .	401	639
1995-10-04 lfsstrc.dtx v3.0d		<code>\sqrt</code> : (DPC) Make robust /1808 . .
General: (DPC) Modify autoload code . . . . .	449	638
1995-10-04 ltab.dtx v1.1d		1995-10-16 ltspace.dtx v1.2j
General: Modify autoload support .	676	<code>\nolinebreak</code> : (DPC) Use <code>\@testopt</code> /1911 . . . . .
1995-10-06 ltfiles.dtx v1.0w		307
<code>\@missingfileerror</code> : Autoload error	341	<code>\nopagebreak</code> : (DPC) Use <code>\@testopt</code> /1911 . . . . .
1995-10-09 lterror.dtx v1.2j		307
General: Modify autoload support .	275	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 . . .	350	733
1995-10-10 ltfssbas.dtx v3.0h		1995-10-17 lclass.dtx v1.0r
<code>\showhyphens</code> : Use <code>\normalfont</code> and make colour safe, and autoloadable . . . . .	424	<code>\@providesfile</code> : Delay definition of <code>\ProvidesFile</code> till lfinal . . . . .
1995-10-10 ltfsdcl.dtx v3.0c		799
<code>\non@alpherr</code> : (DPC) autoload error message . . . . .	483	<code>\ProcessOptions*</code> : Reset <code>\CurrentOption</code> for graphics/1873 . . . . .
1995-10-10 lplain.dtx v1.1r		803
General: Autoload tracing code . . . .	14	1995-10-17 ldirchk.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’ . . . . .	733	4
		1995-10-17 ltfinal.dtx v1.0v
		<code>\@providesfile</code> : reset macro . . . .
		977
		<code>\reserved@b</code> : reset here after the <code>\input</code> above . . . . .
		976
		1995-10-17 lplain.dtx v1.1s
		<code>\reject</code> : Move <code>\supereject</code> to compat file . . . . .
		29
		1995-10-17 ltab.dtx v1.1e
		<code>\@cline</code> : (DPC) Use <code>\@multicnt</code> . .
		699
		<code>\@multispan</code> : (DPC) Macro added.
		699
		1995-10-19 ltfinal.dtx v1.0w
		<code>\@filelist</code> : Move after <code>\reserved@a</code> setting:-) . . . . .
		977

1995-10-20	ltbibl.dtx v1.1k		\@setref: Switch for refundefined renamed . . . . .	605
	\@citex: Removed refundefined flag	777		
	\nocite: Removed refundefined flag	778	\if@multiplelabels: Macro removed	605
1995-10-20	ltclass.dtx v1.0s			
	\@begindocumenthook: Make setting conditional, for autoload version	816	1995-10-25 ltalloc.dtx v1.1b General: General doc improvements	269
1995-10-20	ltfssbas.dtx v3.0i			
	General: (DPC) Modify autoload code, change \undefined . . . . .	401	1995-10-25 ltfloat.dtx v1.1n \endfloatbox: (CAR) macro added: to unify code for double and single versions . . . . .	759
1995-10-20	ltfsstrc.dtx v3.0e		\enddblfloat: (CAR) unify code for double and single versions . . . . .	758
	General: (DPC) Modify autoload code . . . . .	449	\endfloat: (CAR) unify code for double and single versions . . . . .	757
1995-10-22	ltfssbas.dtx v3.0j			
	General: (RmS) New size function macro \genb@sfcnt needs to be disabled at \document. . . . .	401	1995-10-25 ltdxglo.dtx v1.1d General: Doc cleanup . . . . .	773
1995-10-22	ltfsstrc.dtx v3.0f			
	General: Added 'genb' and 'sgenb' size functions to support new DC font naming scheme. . . . .	449	1995-10-25 ltsect.dtx v1.0q \subparagraphmark: Use \let not \def to save space. . . . .	745
1995-10-23	lttab.dtx v1.1f			
	\@settab: (CAR) Ensure that \hightab increases by at most one . . . . .	683	1995-10-27 ltpictur.dtx v1.1f General: Move initialization to kernel from autoload file . . . . .	728
	\@startline: (CAR) Ensure that \nxttabmar is never larger than \hightab . . . . .	681	1995-10-31 ltboxes.dtx v1.0x \finalstrut: Add \nobreak in horiz mode to allow hyphenation. internal/1931 . . . . .	675
	\poptabs: (CAR) Ensure that \curtab is never larger than \hightab . . . . .	684		
	\tabbing: (CAR) Make \hightab consistently a local variable . . .	683	1995-11-01 fontdef.dtx v2.2m General: add \nfss@catcodes for internal/1932 . . . . .	540
1995-10-24	ltfiles.dtx v1.1a			
	\document: Removed multiplelabels switch . . . . .	327	1995-11-01 ltdirchk.dtx v1.0n General: Initialise \@addtolist to \gobble . . . . .	4
	Removed refundefined switch . . .	327		
1995-10-24	ltfssbas.dtx v3.0k			
	\@defaultsubs: macro removed . .	421	1995-11-01 ltfssini.dtx v3.0g General: (DPC) Switch meaning of \addtolist for cfg files . . .	967
	\wrong@fontshape: Make this code inline since it happens only here	420		
1995-10-24	ltmisen.dtx v1.1b			
	\enddocument@kernel@warnings: Changed logic for producing warning messages and removed switch . . . . .	611	1995-11-01 ltfssbas.dtx v3.0m \DeclareFontShape@: (DPC) Test for \relax not \undefined, internal/1933 . . . . .	402
	Use \@refundefined instead of switch . . . . .	611		
1995-10-24	ltxref.dtx v1.1h			
	\@multiplelabels: Switch for multiplelabels removed . . . . .	605	1995-11-01 ltfssini.dtx v3.0g General: (DPC) Switch meaning of \addtolist for cfg files . . .	535
	\newl@bel: Switch for multiplelabels replaced by inline code . . . . .	605		
	\@refundefined: Switch for refundefined replaced . . . . .	604	1995-11-02 ltfssbas.dtx v3.0n \wrong@fontshape: (DPC) Remove extra space with \string for latex/1676 . . . . .	419
			1995-11-02 ltoutenc.dtx v1.7n General: Changed internal name \a to \tabacckludge to protect against redefinition by malicious users. . .	357
1995-11-07	ltlists.dtx v1.0k		\doendpe: Enclosed \setbox0 assignment by a group so that it leaves the contents of box 0 intact. . . . .	654

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. . . . .	361	General: Added <code>\textasciicircum</code> , <code>\textasciitilde</code> , <code>\textbackslash</code> , <code>\textbar</code> , <code>\textgreater</code> and <code>\textless</code> . . . . .	365
Changed <code>\char32</code> to <code>\@xxxii</code> (two tokens less). . . . .	362	Added <code>\textasciicircum</code> , <code>\textasciitilde</code> , <code>\textregistered</code> and <code>\texttrademark</code> . . . . .	359
Replaced octal number 27 by decimal number 23 to protect against the quote character being active. . . . .	362	Added <code>\textbackslash</code> and <code>\textbar</code> . . . . .	358, 369
Replaced some 0's by <code>\z@</code> (faster).	362	Added <code>\textless</code> and <code>\textgreater</code> . . . . .	359, 370
1995-11-10 ltoutput.dtx v1.1s		1995-12-01 ltoutenc.dtx v1.7u	
<code>\@shipoutsetup</code> : Command removed	912	General: Made <code>\SS</code> a Default, rather than having the default point to the OT1 definition. . . . .	359
<code>\@writesetup</code> : Command removed .	912	1995-12-04 ltspace.dtx v1.2k	
In-lined . . . . .	912	<code>\nobreakspace</code> : (Macro added . . . . .	320
1995-11-14 ltclass.dtx v1.0t		1995-12-04 ltspace.dtx v1.2l	
<code>\@unprocessedoptions</code> : Allow empty option . . . . .	817	<code>\xobeysp</code> : (braces added to definition of tilde . . . . .	320
<code>\@loadwithoptions</code> : macro added .	806	1995-12-04 preload.dtx v2.4e	
<code>\LoadClassWithOptions</code> : macro added . . . . .	806	General: Ulrik Vieth. added 12pt OMS and OML preloads /1989 .	560
<code>\RequirePackageWithOptions</code> : macro added . . . . .	806	1995-12-05 ltdefns.dtx 1.2w	
1995-11-17 ltfsbsbas.dtx v3.0m		<code>\@unexpandable@protect</code> : Removed <code>\unexpandable@noexpand</code> as never used. internal/1733 . . . . .	83
<code>\@wrong@font@char</code> : (DPC) Macro added. latex/1676 . . . . .	421	1995-12-05 ltfiles.dtx v1.1c	
<code>\define@newfont</code> : Redefine <code>\typeout</code> latex/1676 . . . . .	414	<code>\document</code> : <code>\ignorespaces</code> added for latex/1933 . . . . .	328
<code>\wrong@fontshape</code> : Support <code>\@wrong@font@char</code> latex/1676 .	419	1995-12-05 ltfloat.dtx v1.1n	
1995-11-17 ltoutenc.dtx v1.7p		<code>\@textsuperscript</code> : Use <code>\ensuremath</code> for latex/1984. . . . .	768
<code>\UseTextSymbol</code> : Support <code>\@wrong@font@char</code> latex/1676 .	354	1995-12-05 ltoutenc.dtx v1.7v	
1995-11-18 ltoutenc.dtx v1.7q		<code>\@inmathwarn</code> : Changed <code>\TextSymbolUnavailable</code> text .	350
<code>\UseTextSymbol</code> : Modify message slightly . . . . .	354	1995-12-06 ltfsbsbas.dtx v3.00	
1995-11-21 fontdef.dtx v2.2n		<code>\nfss@catcodes</code> : Reset hash, for typeouts etc in fd files . . . . .	416
General: Incorporate changed figures, as in plain.tex . . . . .	553	1995-12-07 ltbibl.dtx v1.11	
1995-11-27 ltfsbsbas.dtx v3.0n		<code>\@citex</code> : Restored name of <code>\G@refundefinedtrue</code> . . . . .	777
<code>\nfss@catcodes</code> : Reset hash, for definitions in fd files . . . . .	416	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. . . . .	768
General: documentation fixes . . . . .	750	1995-12-07 ltxref.dtx v1.1i	
1995-11-28 lfsstrc.dtx v3.0g		<code>\@setref</code> : Switch for refundefined restored . . . . .	605
General: documentation fixes . . . . .	449	<code>\G@refundefinedtrue</code> : Renamed (back) from <code>\G@refundefined</code> . . . . .	604
1995-11-28 ltoutenc.dtx v1.7r			
General: Added math mode checks to text commands. . . . .	349		
doc fixes . . . . .	345		
Renamed <code>\@changed@x@err</code> to <code>\TextSymbolUnavailable</code> . . . . .	349		

1995-12-11 ltoutenc.dtx v1.7w		1996-05-21 ltoutenc.dtx v1.7y	
General: Modified \copyright . . . . .	359	General: Corrected error message (CAR) . . . . .	388
1995-12-13 ltdefns.dtx 1.2x		1996-05-21 ltsect.dtx v1.0s	
\:-: Documentation changed. . . . .	100	\@sect: (DPC) Added extra braces for internal/2148 . . . . .	741
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 . . . . .	741
\@iffileonpath: Change argument handling to not require doubled hash. latex/2024 . . . . .	339	\@ssect: (DPC) Added extra braces for internal/2148 . . . . .	744
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 . . . . .	744
\makeglossary: Make no-op after use pr/2048 . . . . .	774	1996-05-23 ltoutenc.dtx v1.7z	
\makeindex: Make no-op after use pr/2048 . . . . .	774	\@strip@args: \expandafter added to match other changes for latex/2133 . . . . .	354
1996-01-20 ltspace.dtx v1.2m		\add@accent: macro added. latex/2133 . . . . .	351
\vspace: Made robust . . . . .	317	\DeclareTextAccent: Reimplemented using \add@accent to save space latex/2133 . . . . .	351
1996-03-25 ltmath.dtx v1.1a		\DeclareTextCompositeCommand: Modified to cope with new \add@accent command: required removal of check for one argument-command . . . . .	352
\@ensuredmath: Macro added for amslatex/2104 . . . . .	641	1996-05-24 ltoutput.dtx v1.1t	
\ensuremath: Reimplement for amslatex/2104 . . . . .	640	\@specialoutput: Check that \@colroom is less than \vsize, indicating that a float has been added . . . . .	903
1996-04-18 ltpage.dtx v1.0i		Cut-off point changed to 1.5\baselineskip . . . . .	903
General: Improve documentation . . . . .	781	\@topnewpage: Cut-off point changed to 2.5\baselineskip . . . . .	902
1996-04-22 ltmiscen.dtx v1.1c		1996-05-25 ltoutput.dtx v1.1u	
General: Improve Documentation . . . . .	608	\@specialoutput: Correct the above check . . . . .	903
1996-04-22 ltspace.dtx v1.2n		1996-06-03 ltmiscen.dtx v1.1d	
General: Documentation Improvements . . . . .	304	\overline: Exchanged the following two code lines so that \dospecials cannot reset the category code of characters handled by \noligs. . . . .	624
1996-04-22 lttab.dtx v1.1g		General: Move setting of verbatim font and \noligs. . . . .	608
\@tabclassz: (DPC) Extra \hskip keeps tabcolsep in empty columns internal/2122 . . . . .	696	\verb: Put setting of verbatim font after \dospecials so that \dospecials cannot reset the	
1996-04-23 ltcnts.dtx v1.1d			
General: Documentation improvements . . . . .	391		
1996-04-24 ltfiles.dtx v1.1e			
\document: (DPC) Reset \AtBeginDocument eg for latex/1297 . . . . .	327		
1996-05-08 lfsstrc.dtx v3.0h			
\math@egroup: Use \bgroup instead of \begingroup to match a kernel change made in 1994!! . . . . .	461		
1996-05-09 lftntcmd.dtx v3.3t			
\check@icr: Default definitions added . . . . .	565		
1996-05-17 fontdef.dtx v2.2o			
General: \@sqrt removed, at last . . . . .	537, 553		
1996-05-17 ltfiles.dtx v1.1f			
\nofiles: added \write to \protected@write for latex/2146	331		
1996-05-18 ltoutenc.dtx v1.7x			
General: Produce error if encoding not found. pr/2054 . . . . .	388		

	category code of characters handled by \noligs. ....	628	\nfss@catcodes: omit \relax as not needed ....	415
1996-06-10	ltboxes.dtx v1.0y \parboxto: (DPC) Changed \endgraf to \par ....	669	1996-07-26 ltfsdcl.dtx v3.0e \init@restore@version: Removed \ifrestore@version switch and replaced by \init@restore@version ....	483
1996-06-10	ltsect.dtx v1.0t \sect: (DPC) Changed \endgraf to \par ....	741	1996-07-26 lfsstrc.dtx v3.0i \init@restore@glb@settings: macro added replacing \if@inmath switch ....	460
1996-06-13	ltdirchk.dtx v1.0r General: documentation improvements mainly from internal/2174 ....	1	1996-07-26 llists.dtx v1.0l \item: Remove unnecessary \global before \minipage... ....	655
1996-06-14	ltab.dtx v1.1h \tabclassz: (DPC) Change both\z@skip to 1sp for latex/2160 ....	696	Remove unnecessary \global before \nobreak.... ....	656
1996-06-22	ltspace.dtx v1.2o General: Documentation of problems added ....	304	1996-07-26 ltmath.dtx v1.1b General: Removed \global before \ignoretrue in various places. .	629
1996-07-10	ltfinal.dtx v1.0y \toks: Free up memory from scratch registers /2213 ....	976	\ignorefalse: put \global into definition ....	609
1996-07-19	ltoutenc.dtx v1.8a \strip@args: Use char 0 not @ as carrier for \lowercase /2197 ..	354	\begin: remove \global before \ignore... ....	615
1996-07-26	ltboxes.dtx v1.0z \if@minipage: put \global into definition ....	670	\end: remove \global before \ignore... ....	618
1996-07-26	ltclass.dtx v1.0u \classoptionslist: made only preamble ....	790	\ignorespacesafterend: user level macro added ....	609
	\unusedoptionlist: made only preamble ....	790	1996-07-26 ltoutput.dtx v1.1v \testfp: remove \global before \test... ....	947
1996-07-26	ltdefns.dtx v1.2y \reargdef: third arg picked up by \yargdef ....	78	\xtryfc: remove \global before \test... ....	920
	\renew@command: use \noexpand instead of \string ....	79	\ztryfc: remove \global before \test... ....	921
	use \relax in place of empty arg	79	General: put \global into definition remove \global before \test... .	895
	\renew@environment: use \relax in place of empty arg ....	80	\clearpage: add number of missing percents ....	898
1996-07-26	ltfloat.dtx v1.1n \endfloatbox: remove unnecessary \global before \minipage... .	759	1996-07-26 ltplain.dtx v1.1t \sh@ft: replace \dimen\z@ by \dimen@	31
	\savemarbox: remove unnecessary \global before \minipage... .	763	1996-07-26 ltsect.dtx v1.0u \starttoc: removed \global before \nobreak.... ....	746
	\setminipage: remove unnecessary \global before \minipage... .	757	\xsect: Removed \global before \nobreak.... ....	742
	\setnobreak: remove unnecessary \global before \nobreak... .	757	1996-07-26 ltspace.dtx v1.2p \if@nobreak: put \global inside definition ....	310
1996-07-26	ltfsbas.dtx v3.0p \DeclareMathSizes: use faster \if test ....	408	1996-07-27 ltfsbas.dtx v3.0q General: \if@inmath switch removed	413
			1996-07-27 ltspace.dtx v1.2q General: Further documentation of problems ....	304

1996-07-27 ltspace.dtx v1.2r		1996-10-21 ltab.dtx v1.1i	
General: Correct documentation of problems . . . . .	304	\array: Use \set@typeset@protect . . . . .	689
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) . . . . .	688
\@xypar: Remove \global before \@ignore . . . . .	764	\multicolumn: Make \multicolumn long (latex/2180) . . . . .	691
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) . . . . .	683
\@afterheading: Removed \global before \nobreak . . . . .	744	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 . . . . .	656
\@Esphack: Remove \global before \@ignore . . . . .	313	\kern... changed to \setbox... . . . . .	656
1996-08-25 lfssbas.dtx v3.0r		Added setting of \clubpenalty and set \nobreakfalse only when necessary . . . . .	656
\nfss@catcodes: Reset the acute, grave and double quote chars as well . . . . .	416	1996-10-23 ltdirchk.dtx v1.0t	
1996-09-21 ltoutput.dtx v1.1w		\@xsect: Replaced \hskip... with \setbox... as used in \@afterheading . . . . .	742
\@writesetup: Added \@parboxrestore and made consequent deletions: wait for the howls of protest . . . . .	912	1996-10-24 ltboxes.dtx v1.1a	
1996-09-25 ltdirchk.dtx v1.0t		\array@parboxrestore: Added local settings of flags: dangerous! . . . . .	669
General: Move ltxcheck to separate file	13	\@iiiminipage: Use it or lose it (@setminpage): Frank will want to lose it . . . . .	671
1996-09-28 ltmiscen.dtx v1.1f		1996-10-24 ltfloat.dtx v1.1p	
\@xobeysp: Moved to ltspace.dtx . . .	623	\@floatboxreset: Added local settings of flags: dangerous! . . . . .	757
1996-09-28 ltspace.dtx v1.2t		\@marginparreset: Added local settings of flags: dangerous! . . . . .	763
\@xobeysp: Moved from ltmiscen.dtx and redefined to use \nobreakspace . . . . .	320	\@xfloat: Added \nодокумент to trap floats in the preamble . . . . .	754
1996-09-29 ltfiles.dtx v1.1g		1996-10-24 ltoutput.dtx v1.1z	
\document: Added disabling of \nодокумент . . . . .	328	\@addtocurcol: Added \nobreak, etc as appropriate . . . . .	926, 930
1996-09-29 ltoutput.dtx v1.1x		\@specialoutput: Added \nobreak as appropriate . . . . .	905
\newpage: Checks for noskipsec and inlabel added . . . . .	899	\@topnewpage: Added \nодокумент to trap \twocolumn in the preamble . . . . .	901
1996-09-29 ltsect.dtx 1.0w		\newpage: Better checks for noskipsec and inlabel added, plus nobreak . . . . .	899
\@noskipsectrue: Added documentation . . . . .	738	1996-10-25 llists.dtx v1.0n	
1996-09-30 ltoutput.dtx v1.1y		\endtrivlist: Change \indent to \leavevmode . . . . .	653
\newpage: Checks for noskipsec and inlabel removed pending further tests . . . . .	899	Reset flags explicitly . . . . .	653
1996-10-04 ltclass.dtx v1.0v		1996-10-25 ltoutput.dtx v1.2a	
\RequirePackageWithOptions: Reset \@unprocessedoptions for /2269	806	\newpage: Reset all flags explicitly . . . . .	899
1996-10-05 ltfiles.dtx v1.1h			
\clubpenalty: Added setting its value . . . . .	326		
1996-10-08 lfntcmd.dtx v3.3u			
\DeclareTextFontCommand: Removed \check@icr when in vmode since it causes various errors (see pr/2157) . . . . .	563		

1996-10-26 ltlists.dtx v1.0o		1996-11-18 ltoutenc.dtx v1.8d	
\endtrivlist: Correct typo . . . . .	653	General: (DPC) lowercase external file names. internal/1044 . . . . .	388
1996-10-27 ltoutenc.dtx v1.8c		1996-11-20 fontdef.dtx v2.2p	
\@strip@args: Removed macro . . . . .	352	General: lowercase fd and enc.def file names /1044 . . . . .	537
General: Added \r A . . . . .	362	1996-11-20 ltvers.dtx v1.0f	
Added		General: Check for old format modified /2319 . . . . .	36
\textasteriskcentered . . . . .	358, 369	1996-11-23 ltoutenc.dtx v1.8e	
Corrected syntax descriptions . . . . .	346	General: Corrected description . . . . .	346
Removed \aa and \AA . . . . .	358, 362, 365	Extended description . . . . .	347
1996-10-28 lplain.dtx v1.1u		1996-11-28 ltvers.dtx v1.0g	
General: (CAR) More doc changes . . . . .	14	General: Check for old format modified /2319 . . . . .	36
\dotfill: Removed math mode . . . . .	31	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) . . . . .	31	1996-12-06 ltfloor.dtx v1.1r	
1996-10-29 lspace.dtx v1.2u		\caption: Call \setminpage if needed. latex/2318 . . . . .	753
\gnewline: Added macro . . . . .	310	1996-12-06 ltfsini.dtx v3.0h	
\@no@lnbk: Macro replaces \lnbk and \@nolnbk . . . . .	308	General: (DPC) Remove *** from messages internal/2338 . . . . .	535
\\: Corrected and rationalised code . . . . .	308	1996-12-17 ltdefns.dtx v1.0w	
\nolinebreak: Reimplemented both using \no@lnbk . . . . .	307	\g@addto@macro: Use \begingroup to save making a mathord . . . . .	102
1996-10-31 lfinal.dtx v1.0z		1996-12-20 ltsect.dtx v1.0z	
General: Added extra \lcode, hoping it does no harm in T1 (pr/1969) . . . . .	967, 974	\@dottedtocline: Added \nobreak for latex/2343 . . . . .	748
1996-10-31 ltlists.dtx v1.0p		1997-01-08 fontdef.dtx v2.2q	
\@trivlist: Added check for missing item in outer list . . . . .	652	General: Use \DeclareMathDelimiter to set delimiter codes . . . . .	546
1996-10-31 ltsect.dtx v1.0y		\mathparagraph: Define using \DeclareMathSymbol . . . . .	555
General: Corrected and tidied documentation; removed long lines . . . . .	737	1997-01-08 ltfiles.dtx v1.1j	
1996-11-03 lplain.dtx v1.1w		\cinclude: reset \deadcycles latex/2365 . . . . .	335
\dotfill: Saved tokens by using \hb@xt@ . . . . .	31	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 . . . . .	631
\@nodocument: Always define \@nodocument in kernel, so that it can be cleared by \document. . . . .	282	1997-02-05 ltdefns.dtx v1.0x	
1996-11-04 ltlists.dtx v1.0q		\g@addto@macro: missing percent /2402 . . . . .	102
\@trivlist: Moved check for missing item: only checked when not inlabel flag is false . . . . .	652	1997-02-21 ltlists.dtx v1.0r	
1996-11-05 ltfiles.dtx v1.1i		\@item: \ifvoid check added for \noindent. latex/2414 . . . . .	656
\nofiles: Standard \if@nobreak test added . . . . .	331	1997-03-21 ltcounts.dtx v1.1e	
1996-11-09 ltmath.dtx v1.1c		\fnsymbol: Use \mathsection and \mathparagraph. latex/2445 . . . . .	396
\@ensuredmath: Made long, as it was before. /2104 . . . . .	641		
1996-11-18 ltfsbas.dtx v3.0s			
\define@newfont: (DPC) lowercase fd file names. internal/1044 . . . . .	415		

1997-04-14 ltfiles.dtx v1.1k		1997-08-29 ltoutenc.dtx v1.9f
\document: Set the document space factor defaults. latex/2404 . . . . .	327	General: Added OT4 encoding, provided by Marcin Woliński. . . . .
\normalsfcodes: Macro added (from patch file) latex/2404 . . . . .	331	
1997-04-14 ltoutput.dtx v1.2b		1997-09-09 ltdefns.dtx v1.2z
\@writesetup: Call \normalsfcodes (from patch file) latex/2404 . . . . .	914	\provide@command: Use \begingroup to avoid generating math ords if used in math mode. pr/2573 . . . . .
Move \label and \index (from patch file) . . . . .	914	81
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 . . . . .	777	\@getcirc: Warn if lines become invisible pr/2524 . . . . .
1997-04-30 ltoutenc.dtx v1.9a		722
General: Changed \textsc to \scshape . . . . .	359	\@picture@warn: Macro added pr/2524 . . . . .
Introduced \textcopyright and modified \copyright . . . . .	359	722
Introduced \textcopyright and modify \copyright . . . . .	360	\@sline: Warn if lines become invisible pr/2524 . . . . .
Modified \textunderscore, removing \mathunderscore . . . . .	359	711
Modified \underscore, removing \mathunderscore . . . . .	360	1997-10-06 lccounts.dtx v1.1f
1997-04-30 ltoutenc.dtx v1.9b		\@Roman: Change \@Roman to be fully expandable, so that the result is written properly to files. . . . .
General: Added \leavevmode to \textunderscore . . . . .	359	396
1997-05-04 ltoutenc.dtx v1.9c		\@slowromancap: Macro added. . . . .
General: Added ‘hex index tabs’ . . . . .	366	396
Added TS1 encoding v2.2.beta . . . . .	372	1997-10-08 ltlogos.dtx v1.1h
1997-05-07 ltoutenc.dtx v1.9d		\LaTeX: Simplify macro (force loading of suitable math fonts once). . . . .
General: Added \leavevmode to \textcompwordmark . . . . .	359	323
1997-05-07 ltspace.dtx v1.2v		1997-10-10 ltclass.dtx v1.0y
\newline: Made completely robust. . . . .	309	\endfilecontents: \@currenvir in banner . . . . .
1997-05-29 lfsstrc.dtx v3.0j		820
General: Replaced \\ by \MessageBreak, as suggested by Donald Arseneau. . . . .	451	\reserved@c not \verb@out to save a csname . . . . .
1997-05-29 ltlogos.dtx v1.1f		819
\LaTeXe: Added \math so that the L <sup>A</sup> T <sub>E</sub> X 2 $\epsilon$ logo works with non-zero values of \mathsurround. . . . .	323	Check for text before or after \end environment. latex/2636 . . . . .
1997-06-16 ltdirchk.dtx v1.0v		820
General: documentation improvements mainly from internal/2520 . . . . .	1	Use \@gobbletwo . . . . .
1997-06-16 ltfloat.dtx v1.1s		819
General: documentation fixes . . . . .	750	1997-10-17 lfntcmd.dtx v3.3w
1997-06-16 lfntcmd.dtx v3.3v		\check@nocorr@: Check for vertical mode moved here, from \DeclareTextFontCommand (see PR/2646). . . . .
General: Fix typo in documentation. . . . .	561	565
1997-08-05 ltoutenc.dtx v1.9e		\DeclareTextFontCommand:
General: Corrected order of arguments in \UseTextSymbol example. . . . .	346	Reinstalled \check@icr as check is now done in \check@nocorr@ (see PR/2646). . . . .
		563
		1997-10-20 ltfinal.dtx v1.1a
		\cuclclist: Removed \aa and \AA from \cuclclist as these are macros. . . . .
		974
		1997-10-21 ltdefns.dtx v1.2z1
		\renew@command: Use \begingroup/\endgroup rather than braces for grouping, to avoid generating empty math atom. . . . .
		79
		1997-10-21 lfssbas.dtx v3.0t
		\define@newfont: Move \makeatletter to \nfss@catcodes. . . . .
		415

\nfss@catcodes: Moved \makeatletter from \try@load@font@shape. . . . .	415	Removed default settings, see next section. . . . .	372
1997-11-09 ltoutput.dtx v1.2c \@specialoutput: Remove incorrect code: only one \emptycol is needed here . . . . .	903	1997-12-19 ltoutenc.dtx v1.9i General: Documentation corrections. 345	345
\@topnewpage: Documentation of vsize check enhanced . . . . .	900	1997-12-20 fontdef.dtx v2.2s General: Added documentation . . . . .	539
1997-11-13 ltfsdcl.dtx v3.0f \DeclareSymbolFont: (DPC) Really update \group@list don't leave new version in \toks@. latex/2661 491	491	1997-12-31 ltoutenc.dtx v1.9k General: Further correction . . . . .	346
\stepcounter: (DPC) Remove as never used. (Re)defined in ltcounts . . . . .	481	1998-01-12 ltoutenc.dtx v1.9k General: Added \ProvidesPackage for textcomp.sty . . . . .	345
1997-11-19 ltfloat.dtx v1.1t \@footnotetext: Missing percent, again . . . . .	769	Adding missing braces and \ushape. . . . .	374
1997-11-19 ltoutput.dtx v1.2d \@vtryfc: Reindent code, to be understandable(DPC). . . . .	920	1998-01-16 ltoutenc.dtx v1.9m General: fixed decimal codes. latex/2734 . . . . .	370
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) . . . . .	484	1998-03-04 ltdefns.dtx v1.2z2 \@xargdef: Unnecessary \expandafter removed: pr/2758 . . . . .	77
\select@group: (DPC) inline use of \stepcounter (faster, and saves a csname per math version as no reset list) . . . . .	482	1998-03-05 ltoutenc.dtx v1.9n General: Added masc/fem ords as in pr/2579 . . . . .	359
1997-11-23 ltoutenc.dtx v1.9g General: Use \textperthousand, \textpertenthousand and \textfractionsolidus not \textpermill, \textperenmill and \textfraction. /2673 . . . . .	372	1998-03-20 ltdefns.dtx v1.2z3 \@thirdofthree: Macro added . . . . .	82
1997-12-17 ltoutenc.dtx v1.9h General: Added \textperthousand and \textpertenthousand . . . . .	363, 364	1998-03-20 ltoutenc.dtx v1.9o General: Documentation added about order of decls . . . . .	348
Added code for textcomp.sty. . . . .	387	Documentation added for pr/2783 . . . . .	347
Added section. . . . .	387	\UndeclareTextCommand: Macro added for pr/2783 . . . . .	356
Added textcomp.sty. . . . .	345	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. . . . .	364	\UndeclareTextCommand declarations for pr/2783 . . . . .	599
Changed to decimal codes in \ooalign. . . . .	374	Load decls after defaults for speed.	599
Changed to decimal codes. . . . .	370	1998-03-21 ltclass.dtx v1.0z General: Added to documentation of	
Documentation changes and additions. . . . .	345	filecontents . . . . .	785
Example corrected, braces removed. . . . .	345	1998-03-21 ltclass.dtx v1.1a \@providesfile: Allow & Internal/2702 . . . . .	799
		General: Correct to new onlypreamble command list . . . . .	834
		1998-03-25 ltfsbas.dtx v3.0u \showhyphens: Suppress unnecessary error when used in preamble . . . . .	424
		1998-04-11 fontdef.dtx v2.2t General: Added \mathring accent (pr2785) . . . . .	553
		1998-04-15 fontdef.dtx v2.2u General: Use new syntax for \DeclareMathDelimiter . . . . .	546

1998-04-15 ltfssdcl.dtx v3.0h \@xxDeclareMathDelimiter: Macro added (pr/2662) . . . . .	501	1998-08-17 ltfnntcmd.dtx v3.3x General: (RmS) Minor documentation fixes. . . . .	561
1998-04-17 fontdef.dtx v2.2v General: Reinsert symbol defs for < and > chars. . . . .	547	1998-08-17 ltfssbas.dtx v3.0v General: (RmS) Documentation fixes. . . . .	401
1998-04-18 fontdef.dtx v2.2w General: Reinsert symbol def for / char. . . . .	547	1998-08-17 ltfssdcl.dtx v3.0i General: (RmS) Corrected minor glitches in changes entries. . . . .	479
1998-05-07 ltclass.dtx v1.1b \load@onefilewithoptions: Modify help message for latex/2805 . . . . .	811	1998-08-17 ltfssini.dtx v3.0i General: (RmS) Minor documentation fixes. . . . .	509
1998-05-18 ltab.dtx v1.1j \@endpbox: Use \setlength to set \hsize, so that the changes in the calc package apply here. . . . .	699	1998-08-17 ltlogos.dtx v1.1i General: (RmS) Minor documentation fixes. . . . .	323
\tabular*: Use \setlength, so that calc extensions apply. . . . .	688	1998-08-17 ltmath.dtx v1.1c General: (RmS) Minor documentation fixes. . . . .	629
1998-05-20 ltfinal.dtx v1.1b General: Set up lccodes before loading hyphenation files: pr/2639 . . . . .	966	1998-08-17 ltmissen.dtx v1.1g General: (RmS) Minor documentation fixes. . . . .	608
Set up uc/lccodes after loading hyphenation files: pr/2639 . . . . .	973	1998-08-17 ltspace.dtx v1.2w General: Documentation fixes. . . . .	304
1998-05-28 lterror.dtx v1.2n \@notdefinable: Added message re 'end...' pr/1555 . . . . .	282	1998-08-17 preload.dtx v2.1g General: (RmS) Minor documentation fixes. . . . .	558
1998-06-04 ltboxes.dtx v1.1c \@rule: Support calc-expressions . . .	673	1998-09-19 ltoutenc.dtx v1.9r \@a: Added \string (pr/2878) . . . . .	357
1998-06-12 ltoutenc.dtx v1.9p General: Corrected 130 and 131, see pr/2834 . . . . .	375	1998-11-13 ltab.dtx v1.1m \@array: Check for hmode to see if something went wrong during parsing (pr/2884) . . . . .	689
Renamed \textmacron pr/2840 . . . . .	376	1999-01-05 fontdef.dtx v2.2x General: Need special protection for character > in \changes entry. . . . .	537
1998-06-12 ltoutenc.dtx v1.9q \add@accent: Explicitly set \spacefactor after \accent (pr/2877) . . . . .	352	1999-01-06 ltfssbas.dtx v3.0w \DeclareFontEncoding: Added \LastDeclaredEncoding to support cyrillic integration (pr/2988) . . . . .	405
1998-06-12 lttextcomp.dtx v1.9p General: Renamed \textmacron pr/2840 . . . . .	595	\LastDeclaredEncoding: Added \LastDeclaredEncoding to support cyrillic integration (pr/2988) . . . . .	405
1998-06-18 ltab.dtx v1.1k General: Small addition to documentation . . . . .	676	1999-01-06 ltoutenc.dtx v1.9r \@strip@args: New impl for latex/2930 . . . . .	354
1998-07-06 ltab.dtx v1.1l General: Small correction to documentation . . . . .	676	General: Minor documentation fix. . . . .	374
1998-08-17 ltboxes.dtx v1.1e General: (RmS) Minor Documentation fixes. . . . .	660	1999-01-06 ltoutput.dtx v1.2e \@makecol: Added negative vskip, as when processing outputbox below: suggested by Fred Bartlett pr/2892 . . . . .	909
1998-08-17 ltclass.dtx v1.1c General: (RmS) Minor documentation fixes. . . . .	785	1999-01-07 ltdefns.dtx v1.3a \@ifnextchar: made long . . . . .	97
1998-08-17 ldirchk.dtx v1.0w General: (RmS) Documentation improvements. . . . .	1		

\@newenvb: made long and brace optional arg. latex/2896 . . . . .	80	1999-06-12 ltoutenc.dtx v1.9v General: Extend \@uclclist only once . . . . .	389
\@testopt: made long and brace optional arg. latex/2896 . . . . .	77	1999-10-09 ltmath.dtx v1.1e \active@math@prime: Macro added, see PR 3104. . . . .	635
1999-01-07 ltdefns.dtx v1.3b \ifnextchar: extra \long. latex/2902 . . . . .	97	\prime@s: Introduce \active@math@prime. . . . .	636
1999-01-07 ltoutenc.dtx v1.9r General: Hackery to allow using fontenc several times . . . . .	390	1999-10-09 ltoutput.dtx 1.2f \@activechar@info: Reset definition of active prime character (used in math mode) . . . . .	912
Hackery to temp support cyrillic uc/lc . . . . .	388	1999-10-28 ltoutenc.dtx v1.9w \add@accent: Give \accent@spacefactor a default definition (pr/3084) . . . . .	352
1999-01-13 ltoutenc.dtx v1.9s \@strip@args: Simplified solution for latex/2930 . . . . .	354	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. . . . .	388
1999-01-18 ltdefns.dtx v1.3c \@yargd@f: New implementation DPC /2942 . . . . .	78	2000-01-07 ltmiscen.dtx v1.1h \verbatim: Disable hyphenation even if the font allows it. . . . .	624
1999-02-09 ltdefns.dtx v1.3d \@yargd@f: catch bad argument forms by re-inserting #3 . . . . .	78	2000-01-15 ltpictur.dtx v1.1i \upvector: Removed space at end-of-line, CAR . . . . .	714
1999-02-12 lttextcomp.dtx v3.0j \legacyoldstylenums: Use \rmdefault instead of cmm (pr/2954) . . . . .	570	2000-01-30 ltfntcmd.dtx v3.3y \DeclareTextFontCommand: Use \hmode@bgroup now (pr/3160) . . . . .	563
1999-02-24 ltoutenc.dtx v1.9t General: Corrected hackery cyrillic uc/lc list . . . . .	388	2000-01-30 ltoutenc.dtx v1.9y General: Use \hmode@bgroup where applicable (pr/3160) . . . . .	361–364, 369–372, 374
1999-03-01 ltdefns.dtx v1.3e \@ifnextchar: remove extra \long. internal/2967 . . . . .	97	\add@accent: Use \hmode@bgroup where applicable (pr/3160) . . . . .	351
1999-04-15 ltpictur.dtx v1.1h \@getlarrow: Replaced octal number, CAR . . . . .	713	\hmode@bgroup: Macro added . . . . .	352
\@upvector: Replaced octal number, CAR . . . . .	714	2000-01-30 ltoutenc.dtx v1.9z \use@text@encoding: Macro reimplemented (pr/3160) . . . . .	354, 355
General: Replaced octal number, CAR . . . . .	713, 714	\add@accent: Macro reimplemented (pr/3160) . . . . .	351
Replaced octal numbers, CAR . . . . .	701	\hmode@start@before@group: Macro added (pr/3160) . . . . .	355
1999-04-19 ltfloat.dtx v1.1u \caption: Made caption an error outside a float: latex/2815 . . . . .	753	2000-05-19 ltmiscen.dtx v1.1i \enddocument: Reset \AtEndDocument for latex/3060 . . . . .	609
1999-04-27 ltboxes.dtx v1.1f \parboxto: (CAR) Changed \empty to \relax as flag for natural width: pr/2975 . . . . .	669	2000-05-26 ltpage.dtx v1.0j \markright: Reimplementation to fix expansion error (pr/3203). . . . .	783
1999-04-29 ltdefns.dtx v1.3f \@yargd@f: Full expansion and conversion needed for digit in new version, see pr/3013 . . . . .	78	\leftmark: Use \empty instead of brace group (pr/3203). . . . .	783
New macro added . . . . .	78		
1999-06-10 ltoutenc.dtx v1.9u General: Ensure that we also forget old options (pr/2888) . . . . .	390		

\markright: Reimplementation to fix expansion error (pr/3203). . . . .	782	(pr/3334) . . . . .	799
\rightmark: Use \empty instead of brace group (pr/3203). . . . .	783	2001-05-25 ladirchk.dtx v1.0x General: Explicitly set catcode of \endlinechar to 10 (pr/3334) . . . . .	4
2000-06-02 ltpage.dtx v1.0k \@markright: Small adjustment to give slightly less expansion, CAR . . . . .	783	2001-05-28 ltoutenc.dtx v1.93 General: Added composites for compatibility with T1, pr/3295 . . . . .	363
\markright: Small adjustment to give slightly less expansion, CAR . . . . .	782	Changed the effect of \.\i, pr/3295 . . . . .	366
Tidied 1.0j reimplementation, CAR . . . . .	782	2001-06-02 fontdef.dtx v2.2y General: Provide default cfg files (pr/3264) . . . . .	556
2000-07-11 ltmiscen.dtx v1.1j \@enddocument@kernel@warnings: Fix typo in warning . . . . .	610	2001-06-04 fontdef.dtx v2.2z General: Guard against math active equal and pipe sign in \models (pr/3333) . . . . .	552
2000-07-12 ltoutput.dtx 1.2g General: Ensure that rule is in \normalcolor . . . . .	953	Guard against math active equal sign in \Relbar (pr/3333) . . . . .	552
2000-07-12 ltoutput.dtx 1.2i \makecol: Removed negative vskip, as it gives unacceptable results when the depth is large: pr/3189 . . . . .	909	2001-06-04 ltclass.dtx v1.1e \@providesfile: But only if it is a char (pr/3334) . . . . .	799
2000-07-19 ltoutput.dtx v1.2h \@writesetup: Reset and restore \@if@newlist for internal/3231 . . . . .	913	2001-06-04 ladirchk.dtx v1.0y General: But only if it is a char (pr/3334) . . . . .	4
2000-08-23 ltfinal.dtx v1.1c General: Fix typo in warning . . . . .	968	2001-06-04 ltpictur.dtx v1.1j \@sline: Don't warn for exactly zero pr/3318 . . . . .	711
2000-08-30 ltoutenc.dtx v1.91 \use@text@encoding: Rearranged but no change to final code, CAR (pr/3160) . . . . .	354	2001-06-04 ltvers.dtx v1.0i General: Check for old format disabled	36
\add@accent: Rearranged but no change to final code, CAR (pr/3160) . . . . .	351	2001-06-05 ltoutenc.dtx v1.94 General: Text composite Commands need kludges for ',' - see tbl1903.lvt . . . . .	363
2000-09-01 ltfinal.dtx v1.1d \errhelp: Set error help empty at very end (pr/449 done correctly). . . . .	976	2001-08-26 ltclass.dtx v1.1f \@providesfile: Readded setting of space char (pr/3353) . . . . .	799
2000-09-24 ltfloat.dtx v1.2b \end@dblfloat: FMi: use output routine to defer float . . . . .	758	2002-02-24 lplain.dtx v1.1x \loggingall: Macro added . . . . .	32
2000-09-24 ltoutput.dtx v1.2b \@doclearpage: FMi: ensure \doclearpage is called again until all floats are output. . . . .	907	\loggingoutput: Macro added . . . . .	32
2000-09-24 ltoutput.dtx v1.2n \addtocurcol: FMi: test for wide float was in wrong place . . . . .	925	\showoutput: Use newly added \loggingoutput . . . . .	32
2001-01-07 ltoutput.dtx v1.2j \@writesetup: And do it in the right macro (pr/3286) . . . . .	913	\tracingall: Use newly added \loggingoutput . . . . .	32
2001-02-16 ltxref.dtx v1.1k \newl@bel: Added an extra group level (PR3250), jlb . . . . .	605	2002-06-16 ltoutenc.dtx v1.95 General: Added \textbardbl (pr/3400) . . . . .	369
2001-05-25 ltclass.dtx v1.1d \@providesfile: Explicitly set catcode of \endlinechar to 10		Added default for \textbardbl (pr/3400) . . . . .	358
		2002-06-17 ltoutenc.dtx v1.95 General: Corrected \c for T1 (pr/3442) . . . . .	364
		Definition of \textexcldown changed (pr/3368) . . . . .	362

Definition of \textquestiondown changed (pr/3368) . . . . .	362	2004-01-04 ltbibl.dtx v1.1p \nocite: Changed error message . . . . .	779
2002-06-18 ltoutenc.dtx v1.95 General: Changed def for \textregistered to avoid small caps (pr/3420) . . . . .	359	2004-01-04 ltoutenc.dtx v1.99c General: More adjustments for ogonek (pr/3532) . . . . .	364
2002-10-01 ltfloor.dtx v1.1v \thempfootnote: Use braces around \itshape to keep font change local (pr/3460). . . . .	767	2004-01-23 ltdefns.dtx v1.1g \newenva: Use kernel version of \ifnextchar (pr/3501) . . . . .	80
2002-10-02 ltfsbas.dtx v3.0x \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)	405	\testopt: Use kernel version of \ifnextchar (pr/3501) . . . . .	77
2002-10-28 ltlists.dtx v1.0s \endtrivlist: Check for math mode (pr/3437) . . . . .	653	\xargdef: Use kernel version of \ifnextchar (pr/3501) . . . . .	77
2002-10-28 ltoutenc.dtx v1.96 General: coding change, to follow bug fix by DEK in plain.tex (pr/3469) . . . . .	362, 371	\xdblarg: Use kernel version of \ifnextchar (pr/3501) . . . . .	99
2002-12-13 ltbibl.dtx v1.1n \citex: Added \leavevmode in case citation is at start of paragraph (pr/3486) . . . . .	777	2004-01-23 ltdefns.dtx v1.3g \kernel@ifnextchar: Added macro (pr/3501) . . . . .	98
2003-01-01 ltftcmd.dtx v3.3z General: Code checked and documentation extended by Chris	563	2004-01-28 ltclass.dtx v1.1g \providesfile: Use kernel version of \ifnextchar (pr/3501) . . . . .	799
2003-05-18 ltbibl.dtx v1.1o \nocite: Check if we are after \document . . . . .	779	2004-01-28 ltvers.dtx v1.0k General: Check for old format made 5 years (pr/3601) . . . . .	36
2003-08-27 ltpictur.dtx v1.1k \bezier: added missing displacement pr/3566 . . . . .	730	2004-02-02 fontdef.dtx v2.3 General: Many things from here on made robust . . . . .	551
\@sline: check for \@linechar being empty pr/3570 . . . . .	711	2004-02-02 ltoutenc.dtx v1.99 General: Added \textbigcircle . . . . .	369
2003-10-13 ltfinal.dtx v1.1e General: Added extra \lccode for \ and \textcompwordmark . . . . .	967	2004-02-04 fontdef.dtx v2.3a General: Added bigtriangle synonyms for stmaryrd . . . . .	549
2003-12-16 ltoutput.dtx v1.2k \makecol: Ensure that \elt has a defined state (pr/3586) . . . . .	909	2004-02-04 ltspace.dtx v1.3 \nobreakdashes: (Macro added . . . . .	319
2003-12-30 ltpictur.dtx v1.1j \getcirc: issue warning if circle size can't be met pr/3473 . . . . .	722	2004-02-06 ltoutenc.dtx v1.99d \inmathwarn: New command added to fix severe bug: pr/3563 . . . . .	349
2004-01-03 ltoutenc.dtx v1.99b General: Added \textogonekcentered (pr/3532) . . . . .	364	2004-02-07 ltoutput.dtx v1.2l \docclearpage: Empty kludgeins box if necessary, pr/3528 . . . . .	907
Added composites for \k (pr/3532) . . . . .	369	2004-02-13 ltoutenc.dtx v1.99e General: Documentation fixes: typos	345
Use \ooalign for \k (pr/3532) . . . . .	364	2004-02-15 ltbibl.dtx v1.1q \cite@ofmt: Added hook with default value \hbox . . . . .	780
		\citex: Changed to use a hook with default value \hbox . . . . .	778
		2004-02-15 ltspace.dtx v1.3a \nobreakdashes: (Added spacefactor setting . . . . .	319
		2004-10-20 ltoutput.dtx v1.2m \makecol: Removed dead code . . . . .	909
		2005-07-27 ltfsdcl.dtx v3.0j \DeclareMathAlphabet: (MH) Make document commands robust . . . . .	493

\DeclareSymbolFontAlphabet: (MH)		\t@st@ic: Use switch \ifmaybe@ic instead of \if@tempswa . . . . .	567
Make document commands robust	506		
\new@mathalphabet: (MH) Make		\enddocument: Use braces around	
document commands robust . . .	494	\input arg (pr/4124) . . . . .	610
\non@alpherr: (MH) Change because		\enddocument: Change of plan: use	
command is now properly robust	483	\@Cinput instead (pr/4124) . . . . .	610
\SetMathAlphabet: (MH) Make		\in@: Simplified thanks to Bruno. . .	479
document commands robust . . .	495	\@ifclasswith: Re-jig definition after	
2005-09-27 ltoutenc.dtx v1.99g		more stringent \in@ test. . . . .	796
General: Replace \sh@ft by		\new@mathversion: (Will) Remove	
\ltx@sh@ft . . . . .	361, 364, 371	\global before \newcount	
2005-09-27 ltplain.dtx v1.1y		(unnecessary and caused etex	
\ltx@sh@ft: New macro . . . . .	31	bug). . . . .	490
\sh@ft: Macro no longer used but left		2012-01-20 ltplain.dtx v2.0b	
for compatibility . . . . .	31	\loggingall: etex tracing if available	32
2005-11-08 ltoutenc.dtx v1.99h		2013-07-07 ltclass.dtx v1.1i	
General: Added \ij and \IJ from		General: Correctly describe how the	
babel. (pr/3771) . . . . .	358, 363, 365	date in \@ifpackagelater is used	788
2005-11-10 ltmath.dtx v1.1g		2014-04-18 ltoutput.dtx v1.1o	
\l: (MH) Fixed potential problem in		General: Handle infinite glue from	
\l (pr/3399). . . . .	636	\enlargethispage (pr/4023) . . . . .	954
General: (MH) Minor documentation		2014-04-24 ltoutput.dtx v1.2n	
fixes. . . . .	629	\f1@tracemessage: Renamed internal	
2006-05-18 ltboxes.dtx v1.1g		trace commands; provide as	
\@parboxto: Ensure \@parboxto holds		package . . . . .	943
the value of \tempdimb not the		2014-04-27 ltfloat.dtx v1.2b	
register itself (pr/3867) . . . . .	669	\end@dblfloat: Inline the code to	
2006-09-13 ltoutput.dtx v1.1m		allow some coexistence with	
General: Ensure that rule is in		packages that hook into	
\normalcolor . . . . .	954	\end@float and do not know	
2007-08-05 ltclass.dtx v1.1h		about the algorithm change . . . . .	758
\@fileswithoptions: Prevent loss of		2014-06-10 ltfloat.dtx v1.2b	
brackets PR/3965 . . . . .	808	\end@dblfloat: missing \fi added .	758
2007-08-06 ltcntrl.dtx v1.0h		2014-12-30 ltfinal.dtx v2.0a	
\@fornoop: Really make defs long .	274	\newmarks: macro added . . . . .	961
2007-08-31 ltfsdcl.dtx v3.0l		\newXeTeXintercharclass: macro	
\SetSymbolFont@: Font warning		added . . . . .	961
changed to info for encoding		2014-12-30 ltffloat.dtx v1.2a	
change (pr/3975) . . . . .	492	\@textsubscript: Command added	
2009-09-24 ltvers.dtx v1.0l		(latexrelease) . . . . .	769
General: Stop checking for old format	36	\textsubscript: Command added	
2009-10-20 ltfsdcl.dtx v3.0m		(latexrelease) . . . . .	768
\in@: More robust thanks to Heiko.	479	2014-12-30 ltfsbas.dtx v3.0y	
2009-10-28 lttextcomp.dtx v1.99k		\mathgroup: move allocation to	
General: Added Latin Modern and		ltplain. . . . .	401
TeX Gyre subsets . . . . .	600	2014-12-30 ltoutput.dtx v1.2m	
2009-11-04 lttextcomp.dtx v1.99l		General: Command updated	
General: Added more Latin Modern		(latexrelease) . . . . .	953
and TeX Gyre subsets . . . . .	600		
2009-12-14 ltntcmld.dtx v3.4a			
\ifmaybe@ic: Macro added . . . . .	566		
\maybe@ic@: Use switch \ifmaybe@ic			
instead of \if@tempswa . . . . .	566		

2014-12-30 ltpplain.dtx v2.0a	
\@e@alloc: macro added .....	19
\@e@alloc@\chardef: macro added ...	18
\@e@alloc@\top: macro added .....	18
\@e@ch@ck: macro added .....	19
\extraffloats: macro added .....	20
\newlanguage: New engine-specific allocation scheme (latexrelease) .	17
2014-12-30 ltspace.dtx v1.3b	
\@C: \@C discards spaces when moving (pr3039)(latexrelease) .....	320
2015-01-03 ltdefns.dtx v1.4a	
\typein: use modified definition in luatex .....	75
2015-01-03 ltdirchk.dtx v1.1	
General: Enable extra primitives when LuaTeX is used .....	3
2015-01-03 ltfinal.dtx v2.0a	
General: Skip resetting codes with Unicode engines .....	973
Unicode data loading added ....	964
2015-01-07 ltvers.dtx v1.0n	
\@check@IncludeInRelease: macro added .....	38
2015-01-08 ltboxes.dtx v1.1h	
\framebox: Make Robust (latexrelease) .....	667
\makebox: Make Robust (latexrelease) .....	661
\parbox: Make Robust (latexrelease) .....	668
\raisebox: Make Robust (latexrelease) .....	674
\rule: Make Robust (latexrelease) .....	673
\savebox: Make Robust (latexrelease) .....	664
2015-01-08 ltdefns.dtx v1.4a	
\MakeRobust: Added macro .....	85
2015-01-08 ltlength.dtx v1.1c	
\setlength: to ensure first length argument is terminated. (latexrelease) .....	399
2015-01-08 ltmath.dtx v1.1h	
\@: Make Robust (latexrelease) .....	636
\@]: Make Robust (latexrelease) .....	636
2015-01-09 ltfsmini.dtx v3.1a	
\em: Allow \emph to produce small caps (latexrelease) .....	529
\eminnershape: macro added (latexrelease) .....	529
2015-01-09 ltspace.dtx v1.1h	
\addpenalty: Donald Arseneau's fix from PR/377703 (latexrelease) ..	316
2015-01-10 ltcounds.dtx v1.1h	
\@fnssymbol: Unse \TextOrMath (latexrelease) .....	397
2015-01-11 ltcounds.dtx v1.1h	
\@stpelt: Reset all within counters in one go (latexrelease) .....	393
2015-01-11 ltfloat.dtx v1.2b	
\@dblfloatplacement: float order in 2-column (latexrelease) .....	760
\@xfloat: Check for valid option (latexrelease) .....	754
\end@dblfloat: float order in 2-column (latexrelease) .....	758
2015-01-11 ltfsbas.dtx v3.0y	
\@DeclareMathSizes: Allow arbitrary units (latexrelease) .....	408
2015-01-11 ltspace.dtx v1.3d	
\@EspHack: Allow hyphenation (Donald Arseneau pr/3498) (latexrelease) .....	313
\@espHack: Allow hyphenation (Donald Arseneau pr/3498) (latexrelease) .....	312
2015-01-14 ltoutput.dtx v1.2n	
\@addtocurcol: float order in 2-column (latexrelease) .....	924
\@addtobblcol: float order in 2-column (latexrelease) .....	935
\@addtonextcol: float order in 2-column (latexrelease) .....	931
\@doclearpage: Empty kludgeins box if necessary, pr/3528 .....	906
float order in 2-column (latexrelease) .....	906
\@startdblcolumn: float order in 2-column (latexrelease) .....	918
\@xtryfc: float order in 2-column (latexrelease) .....	920
\@ztryfc: float order in 2-column (latexrelease) .....	921
2015-01-14 ltspace.dtx v1.3e	
\addpenalty: Avoid adding redundant skips (DPC) .....	316
2015-01-17 ltvers.dtx v1.0m	
\@check@IncludeInRelease: modified with \@currname .....	38
2015-01-19 ltvers.dtx v1.0o	
\@check@IncludeInRelease: Optional argument .....	38
2015-01-20 ltoutput.dtx v1.2m	
\f1@tracemessage: Reset \IncludeInRelease flags .....	945

2015-01-22 ltvers.dtx v1.0p		2015-02-21 ltplain.dtx v2.0e	
General: Preserve any \everyjob material inserted by a loader (.ini file) . . . . .	37	General: Removed autoload code . . .	14
2015-01-23 ltfinal.dtx v2.0b		2015-02-21 lttab.dtx v1.1n	
\newmarks: use reserved count 256 . . . . .	961	General: Removed autoload code . . .	676
\newXeTeXintercharclass: use reserved count 257 . . . . .	961	2015-02-21 ltvers.dtx v1.0r	
2015-01-23 ltplain.dtx v2.0c		General: Removed autoload code . . .	36
\extrafloats: reserve counts 256–265 . . . . .	20	2015-02-21 ltvers.dtx v1.0w	
2015-01-24 ltfinal.dtx v2.0c		\@check@IncludeInRelease: set \@currname empty here (in case \IncludeInRelease input early) . . .	37
General: Skip T1-code entirely with Unicode engines . . . . .	964	2015-02-22 ltfsscmp.dtx v3.0e	
2015-02-03 ltfinal.dtx v2.0d		General: Moved all code into latexrelease - obsolete commands are no longer automatically part of the kernel . . . . .	474
General: Set \lccode for – with Unicode engines . . . . .	966	2015-03-02 ltplain.dtx v2.0f	
2015-02-16 ltoutenc.dtx v1.99m		\@mathgroup@top: macro added . . .	19
General: Added \textcommabelow latex/4414 . . . . .	360	\newlanguage: allow 255 math groups in Unicode engines . . . . .	17
2015-02-16 ltoutenc.dtx v1.99n		2015-03-10 ltplain.dtx v2.0g	
General: Added \textcommabelow . . . . .	361	\hideoutput: macro added . . . . .	34
Added composites for ¢ . . . . .	369	\loggingall: Reorganize to be less noisy . . . . .	32
Added composites for ¢ . . . . .	363	\tracingnone: macro added . . . . .	33
2015-02-16 lttextcomp.dtx v1.99m		2015-03-12 ltoutput.dtx v1.2m	
General: Added lmmt (Heiko Oberdiek) latex/4415 . . . . .	600	General: initialise \dbldeflist again . . . . .	896
2015-02-19 ltvers.dtx v1.0q		2015-03-18 ltfssdcl.dtx v3.0q	
\@check@IncludeInRelease: Swap argument order . . . . .	38	\DeclareSymbolFont: Restrict Symbol fonts to 0–15 . . . . .	491
2015-02-20 ltplain.dtx v2.0d		\document@select@group: Introduce \@mathgroup@top . . . . .	484
\loggingall: Spell commands correctly :-) . . . . .	32	\select@group: Introduce \@mathgroup@top . . . . .	482
2015-02-21 ltdefns.dtx v1.4b		2015-03-26 ltfinal.dtx v2.0d	
General: Removed autoload support . . . . .	73	General: Use renamed unicode-letters.def . . . . .	964
2015-02-21 lterror.dtx v1.2o		2015-04-07 ltfssbas.dtx v3.1a	
General: Removed autoload support . . . . .	275	\wrong@fontshape: Try loading fd file if family has changed . . . . .	419
2015-02-21 ltfiles.dtx v1.1m		2015-04-28 ltfinal.dtx v2.0f	
General: Removed autoload support . . . . .	324	\newXeTeXintercharclass: define \@alloc@intercharclass for compatibility with older xelatex initialization . . . . .	961
2015-02-21 ltfssbas.dtx v3.0z		2015-05-10 ltlists.dtx v1.0t	
General: Removed autoload code . . . . .	401	\doendpe: Explicitly reset \clubpenalty before clearing \everypar; see also pr/0462 and pr/4065 . . . . .	654
2015-02-21 ltfsscmp.dtx v3.0d		2015-06-19 ltfinal.dtx v2.0g	
General: Removed autoload code . . . . .	474	\@alloc@intercharclass@top: Use –1 for first range to get contiguous allocation . . . . .	962
2015-02-21 ltfsdcl.dtx v3.0p			
General: Removed autoload code . . . . .	479		
2015-02-21 lfsstsrc.dtx v3.0k			
General: Removed autoload code . . . . .	449		
2015-02-21 ltoutenc.dtx v1.99m			
General: Removed autoload code . . . . .	345		
2015-02-21 ltoutput.dtx v1.2n			
General: Removed autoload code . . . . .	885		
\f@depth: macro added(latexrelease)	905		
2015-02-21 ltpictur.dtx v1.1k			
General: Removed autoload code . . . . .	701		

\newmarks: Use -1 for first range to get contiguous allocation . . . . .	961	module_warning: Function added . . . . .	54
2015-06-19 lplain.dtx v2.0h		modules: Function modified . . . . .	52
General: delete spurious old definition of \newtoks . . . . .	23	create_callback: Function added . . . . .	63
\@alloc: extra braces in case arguments not single token . . . . .	19	provides_module: Function added . . . . .	53
\newlanguage: Use -1 for first range to get contiguous allocation . . . . .	17	luatexbase: Table added . . . . .	52
2015-06-23 lfinal.dtx v2.0h		2015-10-02 ldirchk.dtx v1.2a	
General: set \patch@level in ltvers rather than in lfinal/ltpatch . . . . .	975	General: Allow backing out of unprefixed names . . . . .	3
2015-06-23 ltvers.dtx v1.0t		2015-10-02 lluatex.dtx v1.0b	
General: set \patch@level in ltvers rather than in lfinal/ltpatch . . . . .	36	General: Fix backing out of TeX code . . . . .	51
2015-08-06 lplain.dtx v2.0i		2015-10-02 lluatex.dtx v1.0c	
\xtrafloats: Add \string in case argument is not an unexpandable primitive . . . . .	20	General: Allow backing out of Lua code . . . . .	51
2015-08-23 ldirchk.dtx v1.2		2015-10-02 lluatex.dtx v1.0e	
General: Do not use luatex prefix . . . . .	3	uninstall: Function added . . . . .	67
2015-08-23 ltvers.dtx v1.0v		2015-10-03 lluatex.dtx v1.0f	
General: Allow negative patchlevel for pre-release . . . . .	37	provides_module: use luatexbase_log . . . . .	53
2015-08-30 lplain.dtx v2.1a		2015-10-27 lplain.dtx v2.1b	
\newinsert: new \newinsert implementation . . . . .	22	\xtrafloats: Use global assignment when switching to extended range . . . . .	20
2015-09-205 ltoutput.dtx v1.3a		2015-11-07 lspace.dtx v1.3f	
General: extended \@freelist . . . . .	895	\@esphack: Only space if there is no space at the end of the hlist latex/4443 . . . . .	312
2015-09-24 lluatex.dtx v1.0a		2015-11-14 lluatex.dtx v1.0g	
call_callback: Function added . . . . .	63	General: Track LuaTeX changes for (new)token.create . . . . .	54
callback.register: Function modified . . . . .	60	2015-11-18 lplain.dtx v2.2a	
callback_descriptions: Function added . . . . .	66	\newlanguage: Extended stream allocation in luatex (0.85) . . . . .	17
\catcodetable@atletter: Macro added . . . . .	48	2015-11-19 lplain.dtx v2.2b	
\catcodetable@initex: Macro added . . . . .	48	\newlanguage: Only extend allocation of write streams (see luatex list) . . . . .	17
\catcodetable@latex: Macro added . . . . .	48	2015-11-27 lluatex.dtx v1.0h	
\catcodetable@string: Macro added . . . . .	48	callback_descriptions: Match test in in-callback latex/4445 . . . . .	66
add_to_callback: Function added . . . . .	64	in_callback: Guard against undefined list latex/4445 . . . . .	66
remove_from_callback: Function added . . . . .	65	2015-11-29 lluatex.dtx v1.0i	
new_attribute: Function added . . . . .	55	General: Declare this as local before used in the module error definitions (PHG) . . . . .	52
disable_callback: Function added . . . . .	66	call_callback: Check name is not nil in error message (PHG) . . . . .	63
in_callback: Function added . . . . .	66	create_callback: Check name is not nil in error message (PHG) . . . . .	63
\newattribute: Macro added . . . . .	47	2015-12-02 lluatex.dtx v1.0j	
\newcatcodetable: Macro added . . . . .	47	General: Adjust hastokens to store the result of tex.hastokens(), not the function (PHG) . . . . .	54
\newluabytecode: Macro added . . . . .	50	Assorted typos fixed (PHG) . . . . .	45
\newluachunkname: Macro added . . . . .	50	Declaration/use of first_head fixed (PHG) . . . . .	53
\newluafunction: Macro added . . . . .	49		
\newwhatsit: Macro added . . . . .	50		
module_error: Function added . . . . .	54		
module_info: Function added . . . . .	54		

Remove nonlocal iteration variables (PHG) . . . . .	45	2016-02-18 ltfssdcl.dtx v3.0r \@DeclarMathDelimiter: Check for delimiter not \delimiter . . . . .	502
Remove unreachable code after calls to error() (PHG) . . . . .	45	\@DeclarMathAccent: Check for mathaccent not \mathaccemt . . . . .	497
2015-12-02 lltuataex.dtx v1.0k		\@DeclarMathRadical: Check for radical not \radical . . . . .	504
General: resolve name and i.description (PHG) . . . . .	61	\@DeclarMathSymbol: Check for mathchar not \mathchar . . . . .	499
call_callback: Give more specific error messages (PHG) . . . . .	63	2016-03-13 lltuataex.dtx v1.0n General: contribute_ filter added . . . . .	59
add_to_callback: Give more specific error messages (PHG) . . . . .	64	insert_ local_ par added . . . . .	59
remove_from_callback: adjust initialization of cb local (PHG) . .	65	2016-03-29 ltpictur.dtx v1.11 \@oval: add setting of line tests . . . . .	723
Give more specific error messages (PHG) . . . . .	65	initialise tests . . . . .	723
create_callback: Give more specific error messages (PHG) . . . . .	63	\@covhorz: use glue not leaders if horizontal line not required . . . . .	726
2015-12-10 ltfinal.dtx v2.0i		\@covvert: use glue not leaders if vertical line not required . . . . .	725
General: Use new common Unicode data loaders . . . . .	964	\if@ovhline: macro added (latex/4452) . . . . .	723
2015-12-18 lltuataex.dtx v1.0l		\if@ovvline: macro added (latex/4452) . . . . .	723
General: Load Unicode data from source . . . . .	48	2016-04-22 ltfinal.dtx v2.0q \e@alloc@intercharclass@top: XeTeX 0.99996 has 4096 char classes not 256 . . . . .	962
2016-01-04 ltfinal.dtx v2.0j		2016-06-19 ltoutenc.dtx v1.99m General: OT1 definition (was duplicate T1 definition) . . . . .	363
General: Do not set up inter character classes for XeTeX . . . . .	964	2016-06-20 ltclass.dtx v1.1j General: don't declare as \@onlypreamble . . . . .	795
\@e@alloc@intercharclass@top: Start allocation at one not three . . .	961	2016-07-29 lplain.dtx v2.2c \extrafloats: use \global \chardef . . . . .	20
2016-01-05 ltfinal.dtx v2.0k		\newinsert: fix for tlb-newinsert-001 . . . . .	22
\@e@alloc@intercharclass@top: Remove duplicated code . . . . .	961	2016-10-02 ltclass.dtx v1.2a \@ifclasswith: Ignore spaces while checking for option clash . . . . .	796
2016-01-05 ltfinal.dtx v2.0l		\ExecuteOptions: Ignore spaces in argument . . . . .	804
General: Correct latexrelease guards . .	964	2016-10-15 ltdirchk.dtx v1.2b General: Require eTEX . . . . .	4
Ensure old definitions for inter-character class toks are available using latexrelease . . . .	964	2016-10-15 lterror.dtx v1.2p General: Require eTEX . . . . .	275
Missing brace . . . . .	964	2016-10-15 ltfinal.dtx v2.0r General: Require eTEX . . . . .	961
2016-01-05 ltfinal.dtx v2.0m		2016-10-15 ltfinal.dtx v2.0s General: Tidy up status of char 127 . . . . .	961
General: Undefine XeTeX classes when using patching an older kernel . . . .	964	2016-10-15 ltfssini.dtx v3.1b General: Require eTEX . . . . .	509
2016-01-05 ltfinal.dtx v2.0p		2016-10-15 lplain.dtx v2.2d General: Require eTEX . . . . .	14
General: Only apply XeTeX change if XeTeX is in use . . . . .	964		
2016-02-11 lltuataex.dtx v1.0m			
General:			
pdf_ stream_ filter_ callback removed . . . . .	60		
process_ rule, [hv]pack_ quality append_ to_ vlist_ filter added . .	59		
read_ cidmap_ file added . . . . .	59		
show_ warning_ message added . .	59		
token_ filter removed . . . . .	59		

2016-10-16 lplain.dtx v2.3a		2017-02-19 ltoutenc.dtx v2.0f
\newlanguage: Allow languages up to 16383 in luatex . . . . .	17	General: add \empty to guard against 3rd argument being empty . . . . .
2016-10-19 lcounts.dtx v1.1j		362
\TextOrMath: Test directly for \protected . . . . .	398	declare composites with empty base for hat and tilde, use same slots for \textasciicircum ans
2016-11-06 lplain.dtx v2.3b		\textasciitilde . . . . .
General: Drop \outer entirely . . . . .	14	377
2016-11-09 ltclass.dtx v2.1b		declare straight quotes using new \remove@tlig command . . . . .
\@fileswithoptions: Improve \ifx tests PR/4497 . . . . .	808	377
2016-11-17 ltluatex.dtx v1.0p		2017-02-22 ltoutenc.dtx v2.0g
General: call_edit added . . . . .	59	General: Fix typo introduced at 2.0f . . . . .
2016-12-03 fontdef.dtx v3.0a		2017-02-24 ltoutenc.dtx v2.0h
General: (DPC) Default to TU encoding for Unicode TeX engines . . . . .	539	General: introduce
\shapedefault: (DPC) Default to TU encoding for Unicode TeX engines . . . . .	543	\DeclareUnicodeAccent . . . . .
2016-12-04 ltoutenc.dtx v2.0a		377
General: Added TU encoding . . . . .	377	\DeclareTextCompositeCommand: add
2017-01-01 ltoutput.dtx v1.3b		check whether the accent command is defined for this encoding . . . . .
General: make fpmin negative so ignored even if float height is negative . . . . .	953	352
2017-01-10 ltfsbas.dtx v3.2a		2017-03-08 ltclass.dtx v1.2c
\showhyphens: Add version of \showhyphens that works with XeTeX . . . . .	424	General: add \@parse@version@dash to support yyyy-mm-dd as well as yyyy/mm/dd . . . . .
2017-01-23 ltoutenc.dtx v2.0b		795
General: Added TU specific commands in ASCII range pr/4500 . . . . .	377	2017-03-09 ltfinal.dtx v2.0t
2017-01-24 ltoutenc.dtx v2.0c		\l@nohyphenation: ensure \l@nohyphenation is defined. . . . .
General: Declare TU composites for i and j . . . . .	377	967
Make \textasteriskcentered U+2217 not U+204E . . . . .	377	2017-03-09 ltmiscen.dtx v1.1m
TeX ligature syntax for xetex and luatex reversed . . . . .	377	\overline: Use \language not \hyphenchar . . . . .
2017-01-24 ltoutenc.dtx v2.0d		624
General: Declare macron composites for YyGg . . . . .	377	\verb: Use \language to stop hyphenation . . . . .
2017-02-12 ltoutenc.dtx v2.0e		628
General: Declare fallback code for \textasteriskcentered . . . . .	377	2017-03-10 ltfiles.dtx v1.1n
2017-02-18 ltluatex.dtx v1.1c		\document: Save language default . . . . .
new_attribute: Parameterize count used in tracking . . . . .	55	327
new_bytocode: Parameterize count used in tracking . . . . .	56	2017-03-10 ltoutput.dtx v1.3c
new_chunkname: Parameterize count used in tracking . . . . .	57	\@writesetup: Reset \language . . . . .
new_whatsit: Parameterize count used in tracking . . . . .	56	913
		2017-03-13 ltdefns.dtx v1.5a
		\-\: Define \- in terms of \hyphenchar . . . . .
		100
		2017-03-27 ltdefns.dtx v1.5b
		\@dischyp: Define \@dischyp after \-\ . . . . .
		100
		2017-03-28 ltluatex.dtx v1.1e
		General: glyph_stream_provider added . . . . .
		60
		2017-03-29 ltboxes.dtx v1.3a
		\arrayparboxrestore: Reset \lineskiplimit . . . . .
		670
		2017-04-05 ltoutenc.dtx v2.0i
		\DeclareTextCompositeCommand:
		Declare accent command if not already declared when declaring a composite . . . . .
		352
		2017-04-10 lplain.dtx v2.3c
		\newlanguage: Correction to code to skip write18 in luatex . . . . .
		17

2017-04-11 ltoutput.dtx v2.4a	\newpage: account for the depth of the last row of the page . . . . .	899	2018-05-11 ltfinal.dtx v2.18	General: Make invalid UTF-8 also safe, for legacy filesystem encodings . . .	971
2017-12-17 ltoutput.dtx v1.4b	\@addtonextcol: fix doc guards . . .	931	2018-05-29 ltclass.dtx v1.2j	\endfilecontents: use \csname not \undefined . . . . .	821
2018-01-06 ltdefns.dtx 1.5c	\@ifundefined: Avoid defining undefined commands to \relax . . .	96	2018-08-11 ltoutenc.dtx v2.0j	General: Provide \guillemetleft and \guillemetright . . . . .	365, 371, 380
2018-02-18 ltclass.dtx v1.2d	\@ifl@ter: Added 0 up front to make bad data come out as 0. . . . .	795	2018-08-18 ltluatex.dtx v1.1h	General: append_to_vlist_filter is exclusive . . . . .	59
	General: Introduce rollback concept	826	2018-08-24 ltfinal.dtx v2.1f	\document@default@language: Add to latexrelease (github/68) . . . . .	968
2018-03-08 ltcnts.dtx v1.1k	\@ifbothcounters: Interface added	394	2018-09-02 ltsect.dtx v1.1b	\dottedtocline: Prevent protrusion ( <a href="https://tex.stackexchange.com/q/172785/10109">https://tex.stackexchange.com/q/172785/10109</a> ) . . . . .	748
	\@removefromreset: Interface added	394	2018-09-24 fontdef.dtx v3.0b	General: Start LR-mode if necessary (git/49) . . . . .	555
	\counterwithin: Interface added . . .	395	2018-09-24 ltmath.dtx v1.2b	\smash: Start LR-mode if necessary (git/49) . . . . .	632
2018-03-24 ltclass.dtx v1.2e	\pkgcls@use@this@release: Use full file name for old release . . . . .	831		\vphantom: Start LR-mode if necessary (git/49) . . . . .	632
2018-03-25 ltfinal.dtx v2.1a	General: default to UTF-8 . . . . .	969	2018-09-24 ltspace.dtx v1.3h	\enspace: Start LR-mode if necessary (git/49) . . . . .	321
	\UseRawInputEncoding: Macro added	970		\leavevmode@ifvmode: Macro added (git/49) . . . . .	321
2018-03-27 ltclass.dtx v1.2f	\endfilecontents: Use full file name for old release . . . . .	820	2018-09-26 ltdefns.dtx v1.5e	\renew@command: Always explicitly generate a space after the csname and not rely on \noexpand to save tokens (git/41) . . . . .	79
2018-04-06 ltfinal.dtx v2.1b	\UseRawInputEncoding: Undo changes to \DeclareFontEncoding@ and definition of \DeclareUnicodeCharacter . . .	970	2018-09-26 ltmiscen.dtx v1.1n	\writefile: Sometimes mask the endline char when writing to files (github/73) . . . . .	614
				\add@percent@to@temptokena: Sometimes mask the endline char when writing to files (github/73) . . . . .	613
2018-04-07 ltfinal.dtx v2.1c	\UseRawInputEncoding: Undefine \inputencodingname . . . . .	970		\protected@file@percent: Sometimes mask the endline char when writing to files (github/73) . . . . .	613
2018-04-08 ltclass.dtx v1.2g	\@ifl@ter: Strip leading spaces from dates. . . . .	795	2018-09-26 ltsect.dtx v1.1c	\addcontentsline: Sometimes mask the endline char when writing to files (github/73) . . . . .	746
2018-04-08 ltclass.dtx v1.2h	\onefilewithoptions: Pass expanded date . . . . .	827	2018-10-10 ltspace.dtx v1.3i	\esphack: Don't introduce breakpoints if @nobreak is true	
2018-04-08 ltfinal.dtx v2.1d	General: Delay full UTF-8 handling to \everyjob . . . . .	971			
2018-04-11 ltcnts.dtx v1.1l					
	\counterwithin: Correct default (issue/38) . . . . .	395			
2018-05-02 ltluatex.dtx v1.1g	General: find_sfd_file removed . . .	58			
	finish_synctex_callback added . . .	59			
	glyph_not_found added . . . . .	60			
	read_sfd_file removed . . . . .	58			
2018-05-08 ltclass.dtx v1.2i	\pkgcls@parse@date@arg: Make suspicious rollback a warning not error: github issue 43 . . . . .	829			

	and after sections . . . . .	312	2019-06-18 lltuatem.dtx v1.1j
2018-10-11	ltmisen.dtx v1.1o		General: finish_ synctex_ callback renamed finish_ synctex . . . . . 59
	\@@sverb: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69) . . . . .	626	font_ descriptor_ objnum_- provider added . . . . . 60
	\@setupverbvisiblespace: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69) 625		make_ extensible added . . . . . 60
	\@verbvisiblespacebox: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69) 625		new_ graf added . . . . . 59
	\asciispace: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69) . . . . .	625	page_ objnum_ provider added . . . . . 59
	\verbatim*: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69) . . . . .	626	process_ pdf_ image_ content added . . . . . 59
	\verbvisiblespace: Provide \verbvisiblespace such that it is usable in normal text (github/70) 625		wrapup_ run added . . . . . 59
	Provide visible space in \verb* also for XeTeX and LuaTeX (github/69) . . . . .	625	
2018-10-21	lltuatem.dtx v1.1i		2019-07-01 ltclass.dtx v1.3a
	\new_luafunction: Function added . .	57	\endfilecontents: Support UTF8 and spaces in filecontents environment file name . . . . . 818
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) . . . . .	778	\IfFileExists: Support UTF-8 . . . . . 337
2018-11-18	ltoutenc.dtx v2.0k		\includeonly: Support UTF-8 . . . . . 332
	General: Provide \Hwithstroke and \hwithstroke . . . . .	382	\set@curr@file: Support UTF-8 . . . . . 336
2018-11-19	ltoutenc.dtx v2.0k		2019-07-09 ltfsbas.dtx v3.2c
	General: Added \Hwithstroke and \hwithstroke . . . . .	364	\DeclareErrorFont: Don't set any \f@... macros . . . . . 418
2018-11-28	ltoutput.dtx v1.4d		2019-07-09 ltfsmini.dtx v3.1c
	\@combinedblfloats: Unbox \@outputbox to preserve boxing level (github/94) . . . . .	918	General: Explicitly set some defaults . . . . . 534
2018-12-30	lttab.dtx v1.1p		2019-08-22 ltxref.dtx v1.11
	\@tabclassz: Add extra \hskip to guard against an \unskip at the start of a c-column cell (gh/102) 696		\labelformat: Commanded moved from variorref.sty . . . . . 606
2019-02-07	ltfilehook.dtx v1.1o		\Ref: Commanded moved from variorref.sty . . . . . 606
	\unqu@tefilef@und: Expand \@filef@und before executing second argument (github/109) . .	845	\refstepcounter: Allow \p@... to have an argument . . . . . 606
2019-02-07	ltfiles.dtx v1.1o		2019-08-27 fontdef.dtx v3.0c
	\@swaptwoargs: Helper macro added .	340	General: Various commands made robust throughout the file . . . . . 548
2019-02-07	ltfsbas.dtx v3.2b		2019-08-27 ltboxes.dtx v1.3b
	\define@newfont: Changed wording of warning (github/107) . . . . .	415	General: Various commands made robust . . . . . 660
			\endfilecontents: Make various commands robust . . . . . 818
			2019-08-27 ltdefns.dtx v1.5f
			General: Make various commands robust . . . . . 101
			\MakeRobust: Make the assignments global as we may need to apply them inside a group . . . . . 85
			2019-08-27 ltfilehook.dtx v1.2b
			\unqu@tefilef@und: Make command robust . . . . . 845
			2019-08-27 ltfiles.dtx v1.2b
			\IfFileExists: Make command robust . . . . . 337

2019-08-27 ltfssbas.dtx v3.2d		2019-09-09 ltfssdcl.dtx v3.0s
General: Make various commands robust .....	401	\DeclareMathSymbol: Allow definition if the math symbol was a command already robust .....
2019-08-27 ltfsdcl.dtx v3.0s		499
\DeclareMathAccent: Make math accents robust .....	497	
\set@mathdelimter: Make math delimiters robust .....	504	2019-09-11 ltclass.dtx v1.3c
2019-08-27 ltfsini.dtx v3.1d		\endfilecontents: Support optional argument for filecontents .....
General: Make various commands robust .....	509	2019-09-14 ltfinal.dtx v2.1h
2019-08-27 ltidxglo.dtx v1.1f		\@ucclist: Expand UTF8 chars when case changing (github/177) 974
General: Make \index and \glossary robust .....	773	2019-09-16 ltxref.dtx v1.1m
2019-08-27 ltlength.dtx v1.1d		General: Correctly revert the \p@... change .....
General: Make various command robust .....	399	2019-09-21 fontdef.dtx v3.0d
2019-08-27 ltlogos.dtx v1.1j		General: Distangle alias (gh/184) 550, 554
\TeX: Make \TeX command robust ..	323	2019-10-02 ltxexpl.dtx v0.0
2019-08-27 ltmath.dtx v1.2c		General: Initial version .....
General: Make various commands robust .....	629	2019-10-02 ltfinal.dtx v2.2
2019-08-27 ltmiscen.dtx v1.1p		General: Load ltxexpl .....
General: Make various commands robust .....	608	2019-10-02 lthuataex.dtx v1.1k
\begin: Make command robust ..	615	General: linebreak_ filter is exclusive 59
\end: Make command robust .....	618	mlist_ to_ hlist is exclusive .... 59
2019-08-27 ltoutput.dtx v1.4e		process_ rule is exclusive .... 59
\@begindvibox: Make \AtBeginDvi robust .....	897	2019-10-07 ltab.dtx v1.1q
2019-08-27 ltpage.dtx v1.0l		\extracolsep: This needs to expand 688
General: Make various commands robust .....	781	2019-10-11 ltfiles.dtx v1.2c
2019-08-27 ltpictur.dtx v1.1m		\set@curr@file: Remove one brace group .....
General: Make various commands robust .....	701	2019-10-11 lfsstrc.dtx v3.0l
Remove several unnecessary \gdef definitions .....	701	\@font@aliasinfo: Added 'alias' size function .....
2019-08-27 ltsect.dtx v1.1d		472
General: Make various commands robust .....	737	2019-10-18 ltclass.dtx v1.3d
2019-08-27 ltspace.dtx v1.3j		\load@onefilewithoptions: Initialize \...-h@k only when loading the package or class (gh/198) .....
General: Make various commands robust .....	304	810
2019-08-27 ltab.dtx v1.1q		2019-10-22 lthuataex.dtx v1.1j
General: Make various commands robust .....	676	General: page_ objnum_ provider and process_ pdf_ image_ content classified data .....
Remove several unnecessary \gdef definitions .....	676	59
2019-08-30 lterror.dtx v1.2q		2019-10-25 ltmiscen.dtx v1.1q
\conditionally@traceoff: Macro added .....	285	\add@percent@to@temptokena: Allow unbalanced conditionals in #1 (gh/202) .....
\conditionally@traceon: Macro added .....	285	613
		2019-10-26 ltfiles.dtx v1.2d
		\@iffileonpath: quote on openin ... 339
		\IfFileExists: don't quote name .. 337
		\IfFileExists@: quote on openin ... 338
		\set@curr@file: remove quotes ... 336
		2019-11-01 ldirchk.dtx v1.3a
		\filename@parse: take last . not first 12
		2019-11-02 ltmiscen.dtx v1.1s
		\@centercr: Make \@centercr robust (gh/203) .....
		620

2019-11-02 ltspace.dtx v1.3k		2020-01-20 ltoutenc.dtx v2.0n
\@normalcr: Make also \@normalcr robust ..... 308		General: fix for gh/251 ..... 360
2019-11-09 ltfiles.dtx v1.2e		2020-01-22 lttextcomp.dtx v1.0b
\set@curr@file: expand and \string before removing quotes ..... 336		\@tc@subst: The overall default is \textcompsubstdefault not \substddefault ..... 573
2019-11-10 ltmiscen.dtx v1.1r		2020-01-25 fontdef.dtx v3.0f
\add@percent@to@temptokena: fix to special comment catcodes (gh/202) ..... 613		General: Load t1enc.def last (gh/255) 539
2019-11-11 ltfiles.dtx v1.2f		2020-01-25 ltoutenc.dtx v2.0m
\@iffileonpath: make \@filef@und match quoting used on \openin . 339		General: Load each encoding file only once (gh/255) ..... 388
2019-11-22 ltoutenc.dtx v2.0l		2020-01-27 ltclass.dtx v1.3g
General: Avoid spurious if fontenc selects LY1 as default encoding (gh/199) ..... 389		\endfilecontents: Fix typo in error message ..... 820
2019-11-29 ltclass.dtx v1.3e		2020-01-28 ltclass.dtx v1.3h
\@pr@videopackage: Protect package info text (gh/52) ..... 798		\endfilecontents: Allow spaces in option string and display only unknown options not the whole option list (gh/256) ..... 818
2019-12-17 fontdef.dtx v3.0e		2020-01-31 ltvers.dtx v1.1e
\mddefault: Set \bfdefault to "b" 542		General: Allow for upcoming format as pre-release 0 ..... 37
\shapedefault: Set \shapedefault explicitly to "n" ..... 543		2020-02-02 ltluatex.dtx v1.1l
\updefault: Set \updefault to "up" 542		General: Add reverselist callback type 62 glyph_ info added ..... 60
2019-12-17 lftntcmd.dtx v3.4c		page_ order_ index added ..... 59
\textssc: Macro added ..... 564		post_ linebreak_ filter is reverselist ..... 59
2019-12-17 lfssbas.dtx v3.2e		create_callback: Provide proper \fallbacks for user-defined callbacks without user-provided default handler ..... 63
\usefont: Don't call \fontseries or \fontshape ..... 410		2020-02-05 lfssini.dtx v3.1g
2019-12-17 lfssini.dtx v3.1e		\DeclareFontSeriesDefault: Clarified error text ..... 511
General: Provide custom series settings a la mweights ..... 510		Corrected misspelled csname (gh/264) ..... 511
\DeclareEmphSequence: Provide \emph sequences ..... 528		2020-02-05 lttextcomp.dtx v2.0n
2019-12-18 ltoutenc.dtx v2.0m		General: Changed the package default to info (gh/262) ..... 590
General: Don't fake \textcompwordmark; take default from T1 instead ..... 359		Ensure we are on a new format (gh/260) ..... 590
\add@accent: Avoid code that breaks \accent ..... 352		2020-02-07 lfssini.dtx v3.1h
2019-12-21 fontdef.dtx v3.0e		\symbol: XeTeX-specific version to avoid bug in maths mode. ..... 531
General: Distangle alias (gh/184) 548–551		2020-02-10 lfssaxes.dtx v1.0c
2020-01-05 ltclass.dtx v1.3f		\fontseries: Switch \if@forced@series added ..... 437
\endfilecontents: Support more write streams in LuaTeX gh/238 818		\fontseriesforce: Switch \if@forced@series added ..... 437
2020-01-11 lfssini.dtx v3.1f		\if@forced@series: Switch \if@forced@series added ..... 437
\rmfamily: Streamlined implementation with hook ..... 523		
\ttfamily: Streamlined implementation with hook ..... 523		
2020-01-20 lfssdcl.dtx v3.0t		
\set@mathdelimiter: fix for gh/251 504		

2020-02-10 ltfssini.dtx v3.1h	<code>\@defaultfamilyhook:</code> Add <code>\@defaultfamilyhook to</code> <code>\normalfont (gh/269) . . . . .</code>	524	<code>\prepare@family@series@update:</code> Drop surplus “m” from <code>\target@series@value (gh/291)</code>	515
	<code>\reset@font:</code> Add <code>\@defaultfamilyhook to</code> <code>\normalfont (gh/269) . . . . .</code>	532	<code>\update@series@target@value:</code> Drop surplus “m” from <code>\reserved@d</code> <code>(gh/291) . . . . .</code>	515, 516
2020-02-10 lttextcomp.dtx v1.0c	General: Use <code>\tabacckludge</code> for tabbing where necessary (gh/271)	576	2020-02-27 ltdefns.dtx v1.5g	<code>\gobblethree:</code> Macro added . . . . .
2020-02-11 fontdef.dtx v3.0g	General: Provide value for <code>\@fontenc@load@list (gh/273) .</code>	539	2020-02-27 ltfssaxes.dtx v1.0d	<code>\series@maybe@drop@one@m:</code> Drop “m” in certain values from a fixed list (gh/293) . . . . .
2020-02-11 ltfssini.dtx v3.1h	General: Provide default value for <code>\@fontenc@load@list (gh/273) .</code>	534	<code>\set@target@series:</code> Drop “m” only in a specific set of values (gh/293)	440
2020-02-11 ltoutenc.dtx v2.0o	General: Update <code>\@fontenc@load@list</code> with option list (gh/273) . . . . .	390	2020-02-27 ltfssbas.dtx v3.2g	<code>\DeclareFontShape@:</code> Only “m” if the series value is a member of a fixed list and issue warning if doing it (gh/293) . . . . .
2020-02-14 ltpictur.dtx v1.1n	<code>\linethickness:</code> Suppress spaces following the declaration (gh/274)	706	2020-03-02 ltexpl.dtx v1.0a	General: Don’t load expl3 if already in the format (gh/295) . . . . .
2020-02-18 ltfssini.dtx v3.1i	<code>\bfseries:</code> Make the <code>\ifx</code> selection outside of <code>\fontseries</code> argument so that it is not done several times	518	2020-03-05 ltexpl.dtx v1.1	General: Load xparse.ltx if <code>\NewDocumentCommand</code> is not defined by expl3.ltx . . . . .
	<code>\mdseries:</code> Make the <code>\ifx</code> selection outside of <code>\fontseries</code> argument so that it is not done several times	519	2020-03-06 ltboxes.dtx v1.3c	<code>\clap:</code> Macro <code>\clap</code> added . . . . .
<code>\prepare@family@series@update:</code> No series auto-update when forced (gh/277) . . . . .	514	2020-03-07 lthuataex.dtx v1.1m	<code>\remove_from_callback:</code> Do not call callback.register for user-defined callbacks . . . . .	
Recognize current family if it is not a “meta” family and auto-update series using <code>\bfdefault</code> (gh/277)	514	2020-03-07 ltmath.dtx v1.2e	<code>\negthickspace:</code> Add <code>amsmath</code> math/text spacing commands to the kernel (gh/303) . . . . .	
2020-02-18 ltmath.dtx v1.2d	<code>\mathindent:</code> Make <code>\mathindent</code> a skip register to match amsmath (gh/252) . . . . .	641	2020-03-07 ltspace.dtx v1.3l	General: Moved <code>\thinspace</code> , <code>\negthinspace</code> and <code>\,</code> to ltmath.dtx (gh/303) . . . . .
	<code>\equation:</code> Separate formula and eqn number by at least a space in fleqn option . . . . .	642	2020-03-19 fontdef.dtx v3.0h	General: Support legacy use of <code>\bfdefault</code> and <code>\mddefault</code> (gh/306) . . . . .
2020-02-20 ltclass.dtx v1.3j	<code>\endfilecontents:</code> Fix missing quotes around file name (gh/284) . . . . .	820	2020-03-19 ltfssdcl.dtx v3.0u	<code>\document@select@group:</code> fix for (gnats/3357) . . . . .
2020-02-24 ltfssbas.dtx v3.2f	<code>\DeclareFontShape@:</code> Drop surplus “m” in series when defining fontshape (gh/289) . . . . .	402	2020-03-19 ltfssini.dtx v3.1k	<code>\DeclareFontSeriesDefault:</code> Support legacy use of <code>\bfdefault</code> and <code>\mddefault</code> (gh/306) . . . . .
2020-02-25 ltfssini.dtx v3.1j	<code>\bf@def@ult:</code> Drop surplus “m” from <code>\bfdef@ult</code> and <code>\mddef@ult</code> (gh/291) . . . . .	521	<code>\maybe@update@bfseries@defaults:</code> Support legacy use of <code>\bfdefault</code> and <code>\mddefault</code> (gh/306) . . . . .	518

\mdseries: Support legacy use of \bfdefault and \mddefault (gh/306) . . . . .	519	2020-04-21 lttab.dtx v1.1r \citabcr: Support calc syntax (gh/152) . . . . .	682
2020-04-06 ltfssini.dtx v3.1m \bf@def@ult: Hook added (gh/306) . . . . .	521	\yargarraycr: Support calc syntax (gh/152) . . . . .	690
\maybe@update@bfseries@defaults: Hook added (gh/306) . . . . .	518	2020-04-22 ltmiscen.dtx v1.1u \@sverb: Drop spaces before \verb delimiter (gh/327) . . . . .	626
\maybe@update@mdseries@defaults: Hook added (gh/306) . . . . .	519	2020-04-22 ltoutenc.dtx v2.0p General: y unicode value in tuenc.def . . . . .	345
2020-04-07 ltclass.dtx v1.3k \IfFormatAtLeastTF: Macro added; also in rollback (gh/168) . . . . .	794	2020-04-29 lttextcomp.dtx v1.0d General: Make all capital accents text commands for hyperref (gh/332) . . . . .	576
\load@onefile@withoptions: Use different method to ignore unprocessed options (gh/22) . . . . .	813, 814	2020-05-02 ltfiles.dtx v1.2g \@include: Support spaces in filenames by enclosing the names of .aux-files in quotes (gh/217) . . . . .	333
\ProcessOptions*: Use different method to ignore unprocessed options (gh/22) . . . . .	803	\includeonly: Get rid of leading and trailing spaces from the filename (gh/217) . . . . .	332
\RequirePackageWithOptions: Use different method to ignore unprocessed options (gh/22) . . . . .	806	Improved support for spaces in filenames (gh/217) . . . . .	332
2020-04-09 ltfloat.dtx v1.2d \textsubscript: Set non-zero baseline (gh/249) . . . . .	769	Pass the filename to \@include by value instead of by reference (gh/217) . . . . .	332
\textsubscript: Set non-zero baseline (gh/249) . . . . .	768	2020-05-05 ltxref.dtx v1.1n \refstepcounter: record the counter name in \currentcounter . . . . .	606
2020-04-13 ltfssdcl.dtx v3.0v \process@table: Small update for speed. . . . .	489	2020-05-06 ltspace.dtx v1.3n General: Made softhyphen active in TU engines . . . . .	322
2020-04-13 ltfssini.dtx v3.1n \init@series@setup: Handling \seriesdefault changes (gh/315) . . . . .	517	2020-05-09 ltdefns.dtx v1.5j \@if@DeclareRobustCommand: Added \DeclareCommandCopy (gh/239) . . . . .	93
\seriesdefault@kernel: Handling \seriesdefault changes (gh/315) . . . . .	535	\DeclareCommandCopy: Added \DeclareCommandCopy (gh/239) . . . . .	91
2020-04-21 ltmath.dtx v1.2f \yeqncr: Support calc syntax (gh/152) . . . . .	639	2020-05-11 ltdefns.dtx v1.5j \dischyp: Do not overwrite \- under LuaTeX . . . . .	100
2020-04-21 ltmiscen.dtx v1.1t \icentercr: Support calc syntax (gh/152) . . . . .	621	2020-05-15 ltdefns.dtx v1.5g \typeout: Allow \par in the argument (gh/335) . . . . .	73
2020-04-21 ltpictur.dtx v1.1o \istackcr: Support calc syntax (gh/152) . . . . .	706	2020-05-19 ltfssaxes.dtx v1.0e \series@maybe@drop@one@m: Need to use \edef (gh/336) . . . . .	441
2020-04-21 ltspace.dtx v1.3m \hspace: Support calc syntax (gh/152) . . . . .	321	2020-05-19 ltfssini.dtx v3.2a \IfFontSeriesContextTF: Macros added (gh/335) . . . . .	526
\newline: Support calc syntax (gh/152) . . . . .	309	2020-05-31 ltmiscen.dtx v1.1u \centering: Added \finalhyphendemerits setting (gh/247) . . . . .	621
\@vspace@calcify: Support calc syntax (gh/152) . . . . .	309		
\@vspace@calcify: Support calc syntax (gh/152) . . . . .	318		
\@vspace@calcify: Support calc syntax (gh/152) . . . . .	315		

\raggedleft: Added \finalhyphendemerits setting (gh/247) . . . . .	622	\date: Don't make the command \long (gh/354) . . . . .	737
\raggedright: Added \finalhyphendemerits setting (gh/247) . . . . .	621	2020-08-01 lltuatem.dtx v1.1p General: new_ graf is exclusive . . .	59
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>≤</sub> basic macros to support package loading . . . . .	68	2020-08-02 lltuatem.dtx v1.1q \newattribute: Move reset to 0 inside conditional . . . . .	47
2020-06-04 ltfinal.dtx v2.2a General: Load ltexpl in ltdefns . . .	976	\newluabytocode: Move reset to 0 inside conditional . . . . .	50
2020-06-05 ltclass.dtx v1.3l \@currnamestack: Added \@expl@pop@filename@@ . . . . .	792	\newluachunkname: Move reset to 0 inside conditional . . . . .	50
Added \@expl@push@filename@@ and \@expl@push@filename@aux@@ . . .	791	\newluafunction: Move reset to 0 inside conditional . . . . .	49
2020-06-05 ltfiles.dtx v1.2h \document: Added hook to load l3backend code . . . . .	326	\newwhatsit: Move reset to 0 inside conditional . . . . .	50
2020-06-10 lltuatem.dtx v1.1n General: Define \@gobble/\@firstofone even for L <sup>A</sup> T <sub>E</sub> X to allow early loading. . . . .	45	2020-08-08 ltclass.dtx v1.3m \endfilecontents: define \q@curr@file directly as the quotes have already been removed (gh/220) . . . . .	819
2020-07-04 ltoutenc.dtx v2.0q General: Implement \remove@tlig in LuaT <sub>E</sub> Xwithout font reloading . . .	377	2020-08-10 lltuatem.dtx v1.1r General: Load lltuatem Lua module during format building . . . . .	50
2020-07-08 ltexpl.dtx v1.2d General: Add a last-minute hook for expl3 . . . . .	69	2020-08-15 ltpictur.dtx v1.2a \@defaultunitsset: Macro added . .	703
2020-07-08 ltfinal.dtx v2.2b General: Add a last-minute hook for expl3 . . . . .	967	2020-08-19 ltdefns.dtx v1.5k \@carcube: Made \long for \NewCommandCopy . . . . .	76
2020-07-27 ltmath.dtx v1.2g \cases: Don't make the command \long (gh/354) . . . . .	633	\robust@command@act: Made \robust@command@act (was \declare@command@copy) more generic . . . . .	89
\matrix: Don't make the command \long (gh/354) . . . . .	633	\ShowCommand: Added \ShowCommand (gh/373) . . . . .	92
\pmatrix: Don't make the command \long (gh/354) . . . . .	633	2020-08-19 ltexpl.dtx v1.2e General: Add \@expl@cs@{thing}@spec@@N for \ShowCommand (gh/373) . . . . .	71
2020-07-27 ltoutenc.dtx v2.0r \@use@text@encoding: Don't make the command \long (gh/354) . . . . .	354, 355	Add \@expl@cs@to@str@@N and \@expl@str@if@eq@@nnTF for \NewCommandCopy (gh/239) . . . . .	71
2020-07-27 ltpage.dtx v1.0m \markright: Don't make the command \long (gh/354) . . . . .	782	2020-08-20 lplain.dtx v2.3d \alloc@: Define \alloc@ in terms of \@alloc . . . . .	21
2020-07-27 ltpictur.dtx v1.1p \linethickness: Don't make the command \long (gh/354) . . . . .	706	2020-08-21 ltclass.dtx v1.3o General: Integration of new hook management interface . . . . .	785
2020-07-27 ltsect.dtx v1.1e \author: Don't make the command \long (gh/354) . . . . .	737	2020-08-21 ltdefns.dtx v1.5l General: Integration of new hook management interface . . . . .	73
		\MakeRobust: Make \MakeRobust produce the same command	

structure as \DeclareRobustCommand . . . . .	85	2020-09-26 ltfinal.dtx v2.2j General: Load first aid file if existing . . . . .	977
2020-08-21 ltexpl.dtx v1.2d General: Dropped unused command . . . . .	68	2020-09-30 ltfssini.dtx v3.2d \maybe@update@bfseries@defaults: \bfdefault@previous not \bfseries@previous (gh/395) . . . . .	518
2020-08-21 ltfiles.dtx v1.2i General: Integration of new hook management interface . . . . .	324	\mdseries: \mddefault@previous not \mdseries@previous (gh/395) . . . . .	519
2020-08-21 ltfinal.dtx v2.2i General: Integration of new hook management interface . . . . .	961	2020-10-01 ltclass.dtx v1.3r \@pr@videopackage: Allow for package substitution . . . . .	798
2020-08-21 ltfssaxes.dtx v1.0g General: Integration of new hook management interface . . . . .	448	2020-10-01 ltsect.dtx v1.1e \addcontentsline: add a fourth argument for better hyperref compatibility . . . . .	746
2020-08-21 ltfsini.dtx v3.2b \bf@def@ult: Integration of new hook management interface . . . . .	521	2020-10-04 ltfiles.dtx v1.2j \@include: Quotes around the aux file name removed, they are not needed and upset BibTeX (gh/400) . . . . .	334
\mdseries@defaults: Integration of new hook management interface . . . . .	524	2020-10-04 lthooks.dtx v1.0d General: Definition \AddToHookNext was supposed to be for \AddToHook vice versa (gh/401) . . . . .	248
\maybe@update@bfseries@defaults: Integration of new hook management interface . . . . .	518	2020-10-08 ltclass.dtx v1.3s \@currnamestack: Added missing 2020/02/02 \IncludeInRelease . . . . .	791
\maybe@update@mdseries@defaults: Integration of new hook management interface . . . . .	519	2020-10-11 ltclass.dtx v1.3t \load@onefilewithoptions: Restore \@currpkg@reqd after finished loading a package file (gh/408). . . . .	812
\reset@font: Integration of new hook management interface . . . . .	532	2020-10-18 ltclass.dtx v1.3t \PassOptionsToClass: Drop path from \input@path (gh/414). . . . .	799
\rmfamily: Integration of new hook management interface . . . . .	523	2020-10-23 ltmscen.dtx v1.1w \enddocument: Make enddocument/afteraux one-time . . . . .	610
\ttfamily: Integration of new hook management interface . . . . .	523	2020-10-26 ltmscen.dtx v1.1x \@kernel@before@enddocument: \enddocument should always start out in vmode (gh/385) . . . . .	612
2020-08-21 ltmscen.dtx v1.1v General: Integration of new hook management interface . . . . .	608	2020-11-09 ltclass.dtx v1.3u \pkgcls@rollbackdate@error: Change help text because the package may have existed then — there is just no rollback data (gh/423). . . . .	832
2020-08-21 ltoutput.dtx v1.4f \@begindivbox: Integration of new hook management interface . . . . .	897	2020-11-09 ltmath.dtx v1.2h \@expl@str@map@function@CNN and \@expl@neg@ignorespace@negmedspace and \@expl@neg@ignorespace@negmedspace and \negthickspace have been only in amsmath, so we need to undefine for rollback (gh/423) . . . . .	635
2020-08-23 ltxref.dtx v1.1o \refstepcounter: add default definition of \@currentcounter . . . . .	606	2020-11-20 ltclass.dtx v1.3u \@currp@th: Macro added . . . . .	790
2020-09-06 ltclass.dtx v1.3q \load@onefilewithoptions: Save \@currpkg@reqd so that we don't lose track of package substitutions. . . . .	812		
2020-09-06 ltdefns.dtx v1.5n \char@if@alph: Macro added . . . . .	99		
2020-09-06 ltexpl.dtx v1.2f General: Add \@expl@str@map@function@CNN and \@expl@neg@ignorespace@negmedspace and for \string@makeletter (gh/386) . . . . .	72		
2020-09-09 ltshipout.dtx v1.0b \_\_shipout\_picture\_overlay:n: Prevent overfull box warnings (gh/387) . . . . .	877		

\@kernel@\currpathstack: Macro added . . . . .	793	2020-12-04 ltfssaxes.dtx v1.0h General: Reorganized the rollback data . . . . .	428
\load@onefile@withoptions: Copy option list to the requested package. . . . .	813	\fontseries: Distangle series and shape update (gh/444) . . . . .	437
\PassOptionsToClass: Copy option list to the requested package. . .	800	\fontshape: Distangle series and shape update (gh/444) . . . . .	445
\ProvidesPackage: Use string comparison instead of \ifx . . .	797	\fontshapeforce: Distangle series and shape update (gh/444) . . . . .	445
2020-11-20 ltcmd.dtx v1.0a General: Initial version derived from xparse.dtx . . . . .	103	2020-12-04 ltfssini.dtx v3.2f General: Adjust start values for series and shape (gh/444) . . . . .	534
2020-11-20 ltfilehook.dtx v1.0d \unqu@tmyfile@und: Move loading to \@input@file@exists@with@hooks and expand \@file@und to avoid getting the wrong file name in the case of a substitution. . . . .	844	2020-12-10 ltbibl.dtx v1.1s \nocite: Delay any \nocite in the preamble instead of raising an error . . . . .	779
2020-11-23 ltshipout.dtx v1.0d \_shipout_execute_cont:: Check for both kernel and user hook (gh/431) . . . . .	867	2020-12-10 ltfssbas.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)	410
\_shipout_execute_main_cont:Nnnn: Check for both kernel and user hook (gh/431) . . . . .	869	2020-12-14 ltclass.dtx v1.3v \@currnamestack: Removed \expl@@@hook@curr@name@push@on .	791
2020-11-24 ltexpl.dtx v1.2g General: Support for roll forward (gh/434) . . . . .	70-72	2020-12-18 ltexpl.dtx v1.2h \@kernel@after@enddocument@afterlastpage: Define kernel \enddocument hooks early . . . . .	68
2020-11-24 ltfilehook.dtx v1.0d General: Support for roll forward (gh/434) . . . . .	842	2020-12-22 ltfssaxes.dtx v1.0h \delayed@merge@font@series: Distangle series and shape update (gh/444) . . . . .	439, 440
2020-11-24 lthooks.dtx v1.0f \_hook_end_document_label_check:: Support for roll forward (gh/434)	211	\delayed@merge@font@shape: Distangle series and shape update (gh/444) . . . . .	446
2020-11-24 ltshipout.dtx v1.0d General: Support for roll forward (gh/434) . . . . .	881	2020-12-22 lfsstrc.dtx v3.0n \selectfont: Execute delayed series and shape updates (gh/444) . . .	453
\AtBeginDvi: Support for roll forward (gh/434) . . . . .	880	2021-01-07 ltfilehook.dtx v1.0e General: Added rollback for this case to avoid spurious errors (part of gh/463) . . . . .	855
2020-11-25 ltdefns.dtx v1.5o \carcube: Added missing latexrelease entry . . . . .	76	\unqu@tmyfile@und: Restore \CurrentFile(Path)(Used) after the input (gh/464) . . . . .	845
2020-12-02 ltluatex.dtx v1.1s General: Fix return value of list callbacks . . . . .	61	2021-01-07 lthooks.dtx v1.0h \_hook_strip_double_slash:w: Assume hook name has at least three nonempty parts (gh/464) .	219
2020-12-03 lfsstrc.dtx v3.0m \selectfont: Install a hook in \selectfont (gh/444) . . . . .	454	\_hook_t1_set:c: Manually define some l3tl commands to work around expl3 changes . . . . .	201
2020-12-04 ltfilehook.dtx v1.0d \undeclare@file@substitution: Don't drop file substitution commands on rollback . . . . .	847		

2021-01-08 ltshipout.dtx v1.0f \__shipout_execute_cont:: Added another kernel hook for more flexibility (cf <a href="https://github.com/pgf-tikz/pgf/issu...">https://github.com/pgf-tikz/pgf/issu...</a> 867)	2021-02-10 ltfloat.dtx v1.2e \@footnotetext: Explicitly run \par at the end of footnote text in preparation for paragraph hooks 769
2021-01-10 ltshipout.dtx v1.0g \@kernel@after@shipout@background: Internal hook \@kernel@after@shipout@background added ..... 871	\document@select@group: fix for (gh/501) ..... 485
\RawShipout: Macro added ..... 871	2021-02-16 ltfloat.dtx v1.2f \footref: \footref added ..... 772
2021-01-19 ltshipout.dtx v1.0h \__shipout_run_firstpage_hook:: Handling of firstpage hook altered 871	2021-02-17 ltoutenc.dtx v2.0t General: Adjust values for \textasteriskcentered To match TS1 definition (gh/502) ..... 382
2021-01-21 ltclass.dtx v1.3w \@kernel@currpathstack: Add empty entry for latexrelease ..... 793	Special definition for \textasteriskcentered when missing in TS1 (gh/502) ..... 373
2021-01-21 ltexpl.dtx v1.3a General: Move xparse rollback code to ltcmd.dtx ..... 71	2021-02-18 ltclass.dtx v1.3x \@fileswithoptions: save raw class option list (gh/85) ..... 808
2021-01-21 ltfinal.dtx v2.2l General: Load glyptounicode.tex for pdfTeX ..... 968	\@remove@eq@value: macro added (gh/85) ..... 801
2021-01-22 ltshipout.dtx v1.0i \__shipout_finalize_box:: Add pre_shipout_filter Lua callback ..... 866	\@use@option: value from unused option list (gh/85) ..... 804
2021-01-24 ltexpl.dtx v1.3a General: Define expl3 hooks conditionally ..... 68	\OptionNotUsed: value from unused option list (gh/85) ..... 801
2021-01-31 ltfilehook.dtx v1.0f \@curr@file@reqd: set \protect to \string gh/481 ..... 848	\PassOptionsToClass: save raw option lists (gh/85) ..... 799
2021-02-03 ltfloat.dtx v1.2e \@savemarbox: Explicitly end with \par (gh/489) ..... 763	2021-02-19 ltoutenc.dtx v2.0u General: Add \textnonbreakinghyphen, \textfiguredash and \texthorizontalbar (gh/404) ..... 362, 366, 381
2021-02-04 ltboxes.dtx v1.4b \color@endbox: Always add the color groups (gh/488) ..... 664	2021-02-25 ltfinal.dtx v2.2m General: Improve speed of ToUnicode everyjob loading code ..... 968
2021-02-08 ltfilehook.dtx v1.0g \unqu@tefilef@und: Undo the internal for robust \InputIfFileExists in rollback (gh/494) ..... 846	2021-03-03 ltclass.dtx v1.3y \endfilecontents: Fix overwrite check for files with UTF-8 (gh/415) ..... 820
2021-02-08 ltmiscen.dtx v1.1y \end: Undo the internals for robust \begin and \end in rollback (gh/494) ..... 619	2021-03-05 ltclass.dtx v1.3z \ProcessOptions*: modify so braces to not give errors (gh/513) ..... 802
2021-02-10 ltboxes.dtx v1.4b \@mpfootnotetext: Explicitly run \par in support for paragraph tagging ..... 672	2021-03-12 ltfiles.dtx v1.2k \IfFileExists@: Allow unbalanced conditionals (gh/530) ..... 338
	2021-03-18 ltcmd.dtx v1.0b General: Use \NewModuleRelease. . 103
	2021-03-18 ltfilehook.dtx v1.0h \__filehook_file_pop_assign:nnnn: Define \g_@_input_file_seq to avoid losing data when rolling back. ..... 842

2021-03-18 ltfssaxes.dtx v1.0i	General: Fix rollforward definition.	438	2021-04-20 ltexpl.dtx v1.3c	\@kernel@after@enddocument@afterlastpage:
2021-03-18 ltfssini.dtx v3.2g	General: Add legacy hook definitions for rollback.	524	Don't empty kernel hooks on rollback	68
2021-03-18 lthooks.dtx v1.0i	\_hook\_end\_document\_label\_check:: Only add top-level if not already there.	210	2021-04-20 ltfilehook.dtx v1.0i	\@curr@file@reqd: Make expand to a string (tracks change in l3kernel)
	Remove the (empty) "top-level" from \currnamestack.	211		848
	General: Use \NewModuleRelease.	199	2021-04-26 ltfssbas.dtx v3.2i	\usefont: Unconditionally switch to the requested font face (gh/444)
2021-03-18 ltvers.dtx v1.1f	\check@IncludeInRelease: Add support for usage in \NewModuleRelease.	38		410
	\new@moduledate: Added \NewModuleRelease.	39	2021-04-26 ltfsstrc.dtx v3.0h	\reset@font: Unconditionally switch to the requested font face (gh/444)
2021-03-19 lttextcomp.dtx v1.0e	General: Use \NewModuleRelease	570		532
2021-03-26 lplain.dtx v2.3e	\@unused: Allocate \inputcheck and \unused early so that they are before expl3 allocates more streams (gh/538)	23	2021-04-26 ltfsstrc.dtx v3.0o	\selectfont: Unset the forced series boolean when reaching \selectfont (gh/444)
2021-03-27 ltclass.dtx v1.4a	\currnamestack: Do not completely roll back if expl3 is loaded.	792		454
2021-04-16 ltvers.dtx v1.1g	\new@moduledate: \NewModuleRelease with the same arguments as \IncludeInRelease.	39	2021-04-29 lthooks.dtx v1.0m	\ActivateGenericHook: Add \ProvideHook etc.
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	329		244
2021-04-18 ltluatex.dtx v1.1t	General: input_level_string added	60	2021-04-29 ltoutenc.dtx v2.0v	General: Add composites for \ae/\AE/\æ/\鏗 (gh/552)
2021-04-18 lplain.dtx v2.3f	\loggingall: 3	32		386
	Drop pre- $\epsilon$ -TeX support	32	2021-05-18 ltclass.dtx v1.4b	\raw@classoptionslist: Initialise to \relax to match \classoptionslist
	\tracingnone: 3	33		790
	Drop pre- $\epsilon$ -TeX support	33	2021-05-24 lcmd.dtx v1.0e	General: Use \msg_... instead of \_kernel_msg...
2021-04-19 lcmd.dtx v1.0d	\cmd_cmd_type_cases:Nnnn: Renamed \cmd_cmd_if_xparse:NTF to \kernel_cmd_if_xparse:NTF for cross-module usage	164		103
			2021-05-24 ltfilehook.dtx v1.0k	General: Use \msg_... instead of \_kernel_msg...
				840
			2021-05-24 lthooks.dtx v1.0n	General: Use \msg_... instead of \_kernel_msg...
				199
			2021-05-24 ltpara.dtx v1.0g	General: Use \msg_... instead of \_kernel_msg...
				296
			2021-05-26 ldefsns.dtx v1.5p	\MakeRobust: Normalize error message in \MakeRobust
				85
			2021-06-03 ltclass.dtx v1.4c	\@kernel@currpathstack: Take care of \@kernel@currpathstack when rolling back/forward.
				793
			2021-06-04 lcmd.dtx v1.0f	General: Normalize various error messages
				166

2021-06-05 ltmiscen.dtx v1.1z		\__hook_hook_gput_code_do:nnn: Do not queue removals (gh/625) . . . . .	212
2021-06-06 ltclass.dtx v1.4c	\@sverb: Normalize error message . . . . .	626	
	\@loadwithoptions: handle raw options for gh/580 . . . . .	806	
	\load@onefile@withoptions: Copy raw options for gh/580 . . . . .	813	
	\PassOptionsToClass: apply \expandafter to raw options for gh/580 . . . . .	800	
2021-06-09 ltclass.dtx v1.4b	\endfilecontents: Use \@latex@note@no@line to display the information . . . . .	819	
2021-06-09 lterror.dtx v1.2r	\@latex@note@no@line: Macros added . . . . .	280	
2021-06-09 ltssbas.dtx v3.2j	\DeclareFontShape@: Improve information message . . . . .	402	
2021-07-08 ltcnts.dtx v1.1m	\counterwithin: New implementation for \counterwithout and \counterwithin . . . . .	394	
2021-07-11 lterror.dtx v1.2s	\PackageNoteNoLine: Provide \ClassNote and \PackageNote . . . . .	279	
2021-07-12 ltclass.dtx v1.4d	\@fileswithoptions: add \unexpanded . . . . .	809	
2021-07-19 ltclass.dtx v1.4e	\@classoptionslist: Drop \onlypreamble . . . . .	790	
	\@ifclasslater: Drop \onlypreamble . . . . .	794	
	\@ifclassloaded: Drop \onlypreamble . . . . .	794	
	\@ifclasswith: Drop \onlypreamble . . . . .	795, 796	
	\@ifl@ter: Drop \onlypreamble . . . . .	795	
	\@pkgextension: Drop \onlypreamble . . . . .	790	
	\@optionlist: Drop \onlypreamble . . . . .	794	
	\@unusedoptionlist: Drop \onlypreamble . . . . .	790	
	\IfFormatAtLeastTF: Drop \onlypreamble . . . . .	794	
2021-07-20 ltcmdhooks.dtx v1.0c	\__hook_patch_DeclareRobustCommand:Nnn: Use \robust@command@chk@safe before \@if@newcommand. . . . .	258	
2021-07-22 lthooks.dtx v1.0o	\__hook_gremove_code:nn: Do not queue removals (gh/625) . . . . .	220	
	\__hook_hook_gput_code_do:nnn: Do not queue removals (gh/625) . . . . .	212	
	\load@onefile@withoptions: Make class/name/after a one-time hook . . . . .	814	
	Make class/name/before a one-time hook . . . . .	813	
	Make package/name/after a one-time hook . . . . .	814	
	Make package/name/before a one-time hook . . . . .	813	
2021-07-23 ltfiles.dtx v1.2n	\include: Make include/name/after a one-time hook . . . . .	334	
	Make include/name/before a one-time hook . . . . .	334	
	Make include/name/end a one-time hook . . . . .	334	
2021-07-27 lthooks.dtx v1.0o	\__hook_clear_next:n: Macro made public . . . . .	236	
	\ClearHookNext: Macro added . . . . .	244	
2021-07-28 ltsect.dtx v1.1f	\contentsline: Pick up four arguments (gh/633) . . . . .	747	
2021-07-30 ltcmd.dtx v1.0d	\__cmd_cmd_type_cases:Nnnnn: Added \@cmd_type_cases:Nnnnn for \NewCommandCopy and \ShowCommand support . . . . .	164	
2021-07-31 ltoutput.dtx v1.4e	\fl@tracemessage: Enable display when doing \tracefloatvals . . . . .	944	
2021-07-31 ltoutput.dtx v1.4g	\ShowFloat: Macro added . . . . .	942	
2021-08-02 lthooks.dtx v1.0o	\ActivateGenericHook: Change name . . . . .	244	
	\DisableGenericHook: Change name	244	
2021-08-07 ltcmd.dtx v1.0g	\__cmd_add_grabber:N: Replicate argument processors for all embellishments (gh/639) . . . . .	127	
	\__cmd_add_type_E:w: Replicate argument processors for all embellishments (gh/639) . . . . .	125	
2021-08-08 ltfinal.dtx v2.2p	\IfPDFManagementActiveTF: Default definition added (gh/640) . . . . .	977	
	\@check@IncludeInRelease: Add error to aid debugging . . . . .	38	
2021-08-11 ltiuatex.dtx v1.1u	General: Define missing local function	52	

2021-08-20 lterror.dtx v1.2t	\@badend: Improve \@badend error message (gh/587) . . . . .	283	2021-09-03 ltoutput.dtx v1.4h	\ShowFloat: Renamed, original name never distributed . . . . .	942
2021-08-20 lhooks.dtx v1.0p	General: Added deprecation warnings for old generic hook commands (gh/638) . . . . .	247	2021-09-06 ltfinal.dtx v2.2q	\@cuclclist: Correctly upper and lowercase \ij and \IJ (gh/658) .	975
	Documentation updates for generic hook commands (gh/638) . . . . .	177	2021-09-06 lhooks.dtx v1.0r	\__hook_hook_gput_code_do:n:nn: Use dedicated conditional (gh/606) .	212
	Renames of generic hook commands (gh/638) . . . . .	207		\__hook_if_execute_immediately:n: Macro added (gh/606) . . . . .	239
	Section on generic hooks added (gh/638) . . . . .	192		\__hook_use_once:n: Clean up after \UseOneTimeHook (gh/606) . . . . .	238
2021-08-25 ltclass.dtx v1.4f	General: Standardise generic hook names (gh/648) . . . . .	785		\__hook_use_once_clear:n: Clean up after \UseOneTimeHook (gh/606)	239
	\load@onefile@withoptions: Declare non-generic package and class hooks . . . . .	814	2021-09-10 ltfssini.dtx v3.2i	\bfseries: Do delayed changes to \bfdefault in a separate macro for better reuse (gh/663) . . . . .	518
2021-08-25 lcmdhooks.dtx v1.0d	\__hook_try_put_cmd_hook:w: Simplify generic hook detection .	255		\DeclareFontSeriesDefault: Do delayed changes to \bfdefault or \mddefault first (gh/663) . . . . .	511
2021-08-25 lfilehook.dtx v1.0l	\unqu@tefilef@nd: Declare non-generic file hooks . . . . .	845	\maybe@update@bfseries@defaults:	Do delayed changes to \bfdefault in a separate macro for better reuse (gh/663) . . . . .	518
2021-08-25 lfiles.dtx v1.2o	\Qinclude: Declare non-generic include hooks . . . . .	335	\maybe@update@mdseries@defaults:	Do delayed changes to \mddefault in a separate macro for better reuse (gh/663) . . . . .	519
	Standardise generic hook names (gh/648) . . . . .	334	\mdseries: Do delayed changes to \mddefault in a separate macro for better reuse (gh/663) . . . . .	519	
2021-08-25 lhooks.dtx v1.0p	\__hook_try_declarng_generic_next_hook:nn: Standardise generic hook names (gh/648) . . . . .	214	2021-09-12 lftntcmd.dtx v3.5a	\check@nocorr@: use \unexpanded to make # safe . . . . .	565
2021-08-27 lcmd.dtx v1.0h	\NewDocumentEnvironment: Check for end-of-environment command .	172	2021-09-12 ltoutenc.dtx v2.0w	General: Move zero skip between i and j for hyphenation (gh/658) . . . . .	363
2021-08-27 lfilehook.dtx v1.0l	\__filehook_file_pop_assign:nnnn: Internal message name changes .	842	2021-09-18 ltpara.dtx v1.0i	\para_end:: Use skip rather than kern as guard. . . . .	300
	\__filehook_file_subst_cycle_error:cN: Use \msg... not \__kernel_msg... .	853	2021-09-26 ltfssdcl.dtx v3.0x	\nfss_init_mv_freeze:N: Macro added for (gh/676) . . . . .	486
2021-08-27 lhooks.dtx v1.0q	General: Internal message name changes . . . . .	242		\c@localalphabets: Counter added for (gh/676) . . . . .	484
2021-08-27 ltpara.dtx v1.0i	General: Internal message name changes . . . . .	302		\document@select@group: Test if we should freeze the version (gh/676)	484
2021-08-30 lcmd.dtx v1.0h	General: Added support for \NewCommandCopy . . . . .	132		\freeze@math@version: Macro added for (gh/676) . . . . .	485
	Added support for \ShowCommand . . . . .	137	2021-09-28 lcmdhooks.dtx v1.0e	\__hook_make_prefixes:w: Make patching of commands a global	

	operation (gh/674) . . . . .	261	variable for freezing math version (gh/676) . . . . .	489
	\_\_hook\_patch\_retokenize:Nnnn:		2021-10-15 ltluatex.dtx v1.1v	
	Make patching of commands a global operation (gh/674) . . . . .	267	General: provide\_ charproc\_ data added . . . . .	60
2021-09-28	lthooks.dtx v1.0s		2021-10-16 ltoutenc.dtx v2.0x	
	\_\_hook\_if\_usable\_use:n: Correct usage of older		\add@accent: Dont set \nspacefactor in math mode gh/643	
	\@@_if_file_hook:wTF (gh/675) 238		. . . . .	352
	\_\_hook\_try\_declaring\_generic\_hook\_split:nNNnn:		2021-10-19 ltpara.dtx v1.0k	
	Correct usage of older		General: Remove content from \text\_everypar:D on rollback . . . . .	303
	\@@_if_file_hook:wTF (gh/675) 215		2021-10-20 ltcmdhooks.dtx v1.0f	
2021-10-14	ltboxes.dtx v1.4c		\_\_hook\_make\_prefixes:w: Correct patching by expansion+redefinition when the macro contains a parameter tokens (gh/697). . . . .	259
	\@mpfootnotetext: Explicitly set		2021-12-02 ltcmd.dtx v1.0i	
	\@currentcounter (gh/687) . . . . .	672	\_\_cmd\_run\_code:: Correct defaults for optional arguments in end-of-environment code (gh/712) 111	
2021-10-14	ltfiles.dtx v1.2p		\_\_cmd\_start\_aux:ccnnnn: Correct defaults for optional arguments in end-of-environment code (gh/712) 111	
	\@include: Warn about use in preamble . . . . .	334		
2021-10-14	ltfloat.dtx v1.2g			
	\@footnotetext: Explicitly set			
	\@currentcounter (gh/687) . . . . .	769		
2021-10-14	ltmath.dtx v1.2j			
	\@eqnse1: Explicitly set			
	\@currentcounter (gh/687) . . . . .	639		
	\eqnarray: Explicitly set			
	\@currentcounter (gh/687) . . . . .	643		
2021-10-15	ltfssdcl.dtx v3.0y			
	\DeclareMathVersion: Initialize			

# 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, <a href="#">H201</a> , X509
\"	r226, r378, r419, r457, r468, r579, r611, r638, r646, r652, r656, r662, r666, r672, r678, r685, r686, r692, r696, r746, r792, r1236, r1254, r1260, r1264, r1270, r1274, r1280, r1286, r1293, r1294, r1300, r1304, r1306, r1413, u416, D171, D198, X510
\#	a62, a75, b6, b14, b465, f696, i341, u403, X493
\\$	a74, b4, b13, f695, r307, r444, r451, r565, r804, r811, D631, D638, X494
\%	a75, a105, a107, a127, b14, b463, f696, r491, r493, u405, D640, D760, G135, S1440, S1441, X495
\&	a74, b5, b13, b464, f695, S376, X496
\'	b485, r227, r379, r421, r455, r465, r581, r591, r597, r599, r602, r604, r612, r618, r624, r626, r629, r631, r639, r643, r650, r654, r659, r664, r667, r669, r676, r681, r682, r689, r694, r697, r747, r794, r813, r815, r816, r817, r820, r822, r823, r824, r826, r827, r1230, r1251, r1258, r1262, r1267, r1272, r1275, r1277, r1284, r1289, r1290, r1297, r1302, r1305, r1313, r1314, r1361, r1362, r1367, r1368, r1379, r1380, r1385, r1386, r1414, r1415, r1429, r1430, r1431, r1432, r1445, r1446, u415, z775, A253, D161, G576, H241, J290, J311, K72, V584, X511
\(	<a href="#">H258</a> , H332
\)	b485, <a href="#">H258</a> , H333
\*	u408, <a href="#">H238</a> , S1121, S1252, S1366, S1442
\+	K72
\,	b399, b401, A506, G576, H7, H8, H40, H154, H156, H159, H184, H207, H208, H224
\-	100, 322, b367, f24, <a href="#">f730</a> , o416, o478, r416, r417, r574, r788, r789, u410, G576, J289, J310, K72, X210, X250
\.	b398, b400, q45, q114, q172, r228, r380, r452, r453, r474, r587, r588, r614, r615, r641, r748, r818, r825, r1235, r1317, r1318, r1327, r1328, r1337, r1338, r1355, r1416, r1417, r1453, r1454, u409, D173, D199
\..default	429
\/	a97, f25, i199, u357, u411, S375
\:	b399, b401, f690, f691, <a href="#">H201</a> , <a href="#">H239</a>
\;	634, b399, b401, A500, H185, <a href="#">H201</a>
\<	r575, r739, u406, G576, K71, K109
\=	r229, r381, r473, r749, r1233, r1307, r1308, r1323, r1324, r1346, r1347, r1348, r1373, r1374, r1399, r1400, r1433, r1434, r1435, r1436, r1451, r1452, r1459, r1460, z775, D179, J290, J311, K71
\>	r572, r740, u407, G576, H213, H228, H229, H239, K71
\?	b398, b400, X511
\@ commands:	
\@set_curr_file:nNN	T353
\@set_curr_file_assign:nnnNN	T353
\@textvisiblespace\meta\{hook}	202
\@next\textvisiblespace\meta\{hook}	202
\@par	286
\@TeXversion	1, 6
\@botlist commands:	
\@botlist:	V1029, V1123, V1282
\@botnum commands:	
\@botnum:	V1003
\@car	75
\@cdr	75
\@citeb commands:	
\@citeb:	Q16, Q45, Q62
\@colht commands:	
\@colht:	V541, V555
\@colnum commands:	
\@colnum:	V1188, V1356
\@cons	75
\@currdir	1, 6
\@currname commands:	
\@currname:	q664
\@dblarg	74
\@dbldeflist commands:	
\@dbldeflist:	V1755, V1797
\@dbltopnum commands:	
\@dbltopnum:	V1654, V1781
\@deferlist commands:	
\@deferlist:	V1114, V1211, V1272, V1380, V1424, V1453, V1511, V1543, V1629, V1669

**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=ltspacedtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

```

\@ifclasslater 788 \^ ... a63, a72, a75, a119, a330, b7, b9,
\@ifclassloaded 788 b11, b14, b404, b405, b419, b420,
\@ifclasswith 788 f20, f696, g6, g7, g1712, g1972,
\@ifdefinable 74 g1977, g2652, o480, o482, o484,
\@ifnextchar 74 r231, r286, r382, r456, r466, r558,
\@ifpackagelater 788 r644, r651, r655, r660, r665, r670,
\@ifpackageloaded 788 r677, r683, r684, r690, r695, r750,
\@ifpackagewith 788 r1231, r1248, r1252, r1259, r1263,
\@ifstar 74 r1268, r1273, r1278, r1285, r1291,
\@ifundefined 74 r1292, r1298, r1303, r1315, r1316,
\@missingfileerror 789 r1333, r1334, r1341, r1342, r1356,
\@namedef 74 r1357, r1358, r1387, r1388, r1409,
\@nameuse 74 r1410, r1411, r1412, u398, u399,
\@restorepar 286 u404, D169, D197, G127, G136,
\@setpar 286 S1122, S1123, S1124, S1205, S1208,
\@topnum commands: S1211, S1253, S1254, S1255, S1337,
\@topnum: V1049 S1340, S1343, S1367, S1368, S1369,
\[..... u412, z43, S1427, S1430, S1433, X241, X242,
 z54, z76, z87, H276, H334, H432, X512 X243, X244, X245, X246, X247,
\\ 304 X248, X249, X498, X504, X505,
\\ a45, a46, a74, a247, a248, a249, X506, X507, X541, X542, X543,
 a250, a253, a260, a261, a262, a263, X544, X545, X546, X547, X548, X549
 a266, a273, a274, a275, a276, a279, _ a75, b8, b14,
 a286, a292, a293, a297, a299, a300, f696, r314, A254, H256, H257, X499
 a304, a309, a310, a313, a319, b13, \` r232, r383, r420, r454, r464, r580,
 d264, d417, f219, f276, f437, f525, r642, r649, r653, r658, r663, r668,
 f543, f695, g399, g463, g617, g629, r675, r679, r680, r688, r693, r751,
 g660, g1009, g1015, g2241, g2271, r793, r1229, r1250, r1257, r1261,
 g2285, g2292, g2321, g2411, g2451, r1266, r1271, r1276, r1283, r1287,
 g2462, g2488, g2494, g2501, h893, r1288, r1296, r1301, u414, z775,
 h904, h1314, h1320, h1330, h1350, D175, G576, J290, J311, K72, X514
 h1361, h1362, h1367, h1372, h1377, \| r561, s169, s180, A573, A574, X515
 h1393, h1519, h1520, h1521, i338, \~ a75, b10, b14, f696, l20,
 i397, l277, n104, n111, n118, n119, o422, r239, r287, r384, r467, r559,
 n120, n121, o52, o480, q657, q672, r645, r657, r661, r671, r687, r691,
 r560, u400, A251, G358, G363, r752, r1232, r1249, r1253, r1265,
 G368, G378, G382, G386, G410, r1269, r1279, r1295, r1299, r1343,
 H357, H493, J322, J472, J474, r1344, r1345, r1397, r1398, D187,
 K73, K170, K180, K194, L139, X497 D207, G507, G522, G537, G580, X502
\{ a3, a7, a74, b2, b13, \| a74, a91,
 g548, g1676, i339, l22, r308, r562, b13, b404, b422, f695, l19, l20, l21,
 u401, A249, G409, H59, H154, X500 l22, l25, o421, u397, u647, u683,
\} a8, a74, b3, b13, i340, l21, r309, u708, A252, G406, G407, G512,
 r563, u402, A250, G409, H59, X501 G527, M36, M38, Q17, S369, X492
\langle addto-cmd\rangle 178
\langle cmd\rangle_u 133
\langle cmd\rangle_u(arg_u(num)) 135
\langle cmd\rangle_uu 133
\langle cmd\rangle_u code 133
\langle cmd\rangle_u defaults 133
\langle filename\rangle 850
\langle function\rangle 153
\] ... b485, u413, H276, H335, H456, X513
\A X238, X517, X538
\a r223, K1, X229, X518, X529
\AA b410, r240, r428, r526
\aa b410, r245, r422, r536
\abovedisplayshortskip b385, H501
\abovedisplayskip b384,
 H494, H496, H498, H499, H500, H501

```

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
X=ltfinal.dtx

\accent ... 441, r73, r393, r423, r479, r763  
 \ActivateGenericHook ..... 177, 192, h1401, h1407  
 \active ..... a64, a119, a330, b10, b11, b419, b420, b422, G406, G407, G506, G512, G521, G527, G536, G578, H241, H256, S1122, S1123, S1124, S1205, S1208, S1211, S1253, S1254, S1255, S1337, S1340, S1343, S1367, S1368, S1369, S1427, S1430, S1433, V584  
 \acute ..... A516  
 add commands:  
     add\_to\_callback ..... d707  
 \add\_to\_callback ..... 44  
 \addcontentsline 613, N70, N80, N159, O16  
 \AddEverypageHook ..... 865  
 \addpenalty ..... 316, o270, N50, V338, V1153, V1319, I124, I170, I175  
 \AddThispageHook ..... 865  
 \addtocontents 613, N164, N171, N177, N180  
 \addtocounter ..... 391  
 \addtocounter ..... s6, s18  
 \AddToHook ..... 176, 178, 180, 184, 188, 191–193, 195, 203, 204, 239, 251, 255, 838, 863–865, h1410, h1540, w152, G38, G39, G303, G304, G305, G306, Q52, S1027, S1028, S1029, T524, T526, T536, T537, T538, T539, U188, U462, U485  
 \AddToHookNext ..... 179, 180, 184, 188, 234, 856, 861, 864, 865, h1412, h1538, T525, U486  
 \addtolength ..... 399  
 \addtolength ..... t16, H496, H498  
 \addtoversion ..... x20, x139  
 \addvspace ..... o233, G336, N50, I124, I171, I172, I176, I224  
 \adjdemerits ..... b332  
 \AE ..... r241, r398, r527, r768, r1119, r1429, r1433, X570  
 \ae ..... r246, r401, r537, r772, r1125, r1431, r1435, X570  
 \afterassignment ..... 86, b435, b438, f256, f262, f305, r212, r220, u328, H186  
 \AfterEndEnvironment ..... 195, G299  
 \aftergoup ..... 485  
 \aftergroup ..... u91, u342, w202, w268, y114, y121, y129, C64, G510, G525, G540, J148, V604, V605, V662, V663  
 \AfterLastShipout ..... T536  
 \afterpreamble ..... 328  
 \aleph ..... A308  
 \allocationnumber ..... 298, 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, d205, n39, K4, K9, X44, X45, X46  
 \allowbreak b442, f772, f773, f792, f794, H40  
 \Alph ..... 391  
 \Alph ..... 862, s140  
 \alph ..... 391  
 \alph ..... s139  
 \alpha ..... A268  
 \amalg ..... A379  
 \AmSfont ..... 438  
 \and ..... 737  
 \and ..... N14, N27  
 \angle ..... A337  
 \approx ..... A422  
 \arabic ..... 391  
 \arabic ..... 391, 862, s79, s87, s136, M33, U348  
 \arccos ..... H13  
 \arcsin ..... H10  
 \arctan ..... H16  
 \arg ..... H26  
 \ArgumentSpecification ..... 161, g2060, g2067, g2083, g2090  
 array (environment) ..... K168  
 \array ..... K168  
 \arraycolsep ..... H360, H361, H506, H507, K258, K338  
 \arrayrulewidth .. K324, K338, K346, K347, K359, K363, K366, K376, K378  
 \arraystretch ..... K186, K187, K342  
 \Arrowvert ..... A569  
 \arrowvert ..... A567  
 \asciispace ..... G466, G468, G471, G474, G475, G494  
 \ast ..... A232, A395  
 \asymp ..... A449  
 \AtBeginDocument 178, 182, 191, 193, 196, 447, 779, q125, q180, v737, Q34, S1018  
 \atbegindocumenthook ..... 328  
 \AtBeginDvi ..... 193, 861, 864, 872, 873, 880, U401, U437, V86  
 \AtBeginEnvironment ..... 195, G299  
 \AtBeginShipout ..... 864, U442, U485  
 \AtBeginShipoutAddToBox ..... 863, U490  
 \AtBeginShipoutAddToBoxForeground ..... 863, U490  
 \AtBeginShipoutBox ..... 863, U483  
 \AtBeginShipoutDiscard ..... 864, U489  
 \AtBeginShipoutFirst ..... 864, U445, U487  
 \AtBeginShipoutInit ..... 864, U484  
 \AtBeginShipoutNext ..... 864, U443, U485

**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=ltCMdHOOKS.dtx, j=ltALLOC.dtx, k=ltCNTRL.dtx, l=ltERROR.dtx, m=ltPAR.dtx, n=ltPARA.dtx, o=ltSPACE.dtx, p=ltLOGOS.dtx, q=ltFILES.dtx, r=ltOUTENC.dtx, s=ltCOUNTS.dtx, t=ltLENGTH.dtx, u=ltFSSbas.dtx, v=ltFSSaxes.dtx, w=ltFSStrc.dtx, x=ltFSScmp.dtx, y=ltFSSdcl.dtx, z=ltFSSini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltFNTcmd.dtx, D=ltTEXTcomp.dtx, E=ltPAGENO.dtx, F=ltXREF.dtx, G=ltMISCEN.dtx, H=ltMATH.dtx, I=ltLISTS.dtx, J=ltBOXES.dtx, K=ltTAB.dtx, L=ltPICTUR.dtx, M=ltTHM.dtx, N=ltSECT.dtx, O=ltFLOAT.dtx, P=ltIDXGLO.dtx, Q=ltBIBL.dtx, R=ltPAGE.dtx, S=ltCLASS.dtx, T=ltFILEHOOK.dtx, U=ltSHIPOUT.dtx, V=ltOUTPUT.dtx, W=ltHYPHEN.dtx, X=ltFINAL.dtx

\AtBeginShipoutOriginalShipout 864, U498  
 \AtBeginShipoutUpperLeft .... 863, U490  
 \AtBeginShipoutUpperLeftForeground ..  
 ..... 863, U490  
 \AtEndAfterFileList ..... T538  
 \AtEndDocument 193, 197, G69, S1018, T542  
 \AtEndDvi ..... 861, 864, U452, U457  
 \AtEndEnvironment ..... 195, G299  
 \AtEndOfClass ..... 837, H431, S1018  
 \AtEndOfPackage ..  
 ... 837, S541, S560, S658, S669, S1018  
 \AtEndPreamble ..... 196  
 \AtNextShipout ..... 864  
 \atopwithdelims ..... H57, H58, H59  
 \attribute ..... 41, d79  
 \attributedef ..... d79, d215  
 \attributezero ..... d215  
 \AtVeryEndDocument ..... T537  
 \AtVeryVeryEnd ..... T539  
 \author ..... 737  
 \author ..... N8, N24, N32

**B**

\b ..... r233, r389, r475, r759, r1240  
 \backslash ..... A251, A590  
 \bar ..... A520  
 \baselineskip ..... b403, b433,  
 b469, w186, w187, w188, w190,  
 w191, A510, H158, H159, H178,  
 H184, H188, J299, J318, K198,  
 L136, L314, L373, U220, U271,  
 V242, V273, V622, V637, V681, V696  
 \baselinestretch ..... u319,  
 w120, w121, w166, w167, w184, w245  
 \batchmode ..... 342, e89, e95, e105,  
 q602, q633, q634, x106, X590, X611  
 \BeforeBeginEnvironment ..... 195, G299  
 \BeforeClearDocument ..... T540  
 \begin ..... 85, 111, 135, 176, 194–197,  
 219, 326, 617, g1213, g1319, h1429,  
 i65, l246, l248, q340, w7, A4, A102,  
 B4, G167, G168, H436, H448, L461,  
 N14, N17, S679, U383, W3, X475, X479  
 \begingroup ..... 326, 352, 617  
 \belowdisplayshortskip ..... b387, H500  
 \belowdisplayskip ..... b386, H499  
 \beta ..... A269  
 \bezier ..... 701  
 \bezier L682, L683, L805, L806, L821, L823  
 \bfdefault ..... 199, 441, 510, 511,  
 513, 514, 516, 518, 521, z15, z261,  
 z267, z268, z269, z270, z310, z311,  
 z312, z313, z322, z358, z382, z401,  
 z442, z476, A92, A104, A106, A114

\bfseries ..... 199, 510, 514, 525, z13, z14,  
 z253, z254, z302, z307, z308, z349,  
 z352, z353, z378, z380, z381, z474,  
 z475, C19, F13, M36, M38, Q20, U384  
 bfseries ..... z420  
 bfseries/defaults ..... z420  
 \bgroup ..... b417  
 \bibcite ..... Q7, Q9, Q10  
 \bibdata ..... Q25, Q29  
 \bibitem ..... Q3  
 \bibliography ..... 776  
 \bibliography ..... Q27  
 \bibliographystyle ..... 776  
 \bibliographystyle ..... Q32  
 \bibstyle ..... Q25, Q37  
 \Big A623, A626, A635, A637, H44, H45, H46  
 \big ..... A624, A636, H41  
 \bigbreak ..... b449, f774, f795  
 \bigcap ..... A345  
 \bigcirc ..... A392  
 \bigcup ..... A346  
 \Bigg ..... A630, A639, H50, H51, H52  
 \bigg ..... A628, A638, H47, H48, H49  
 \Biggl ..... H50  
 \biggl ..... H47  
 \Biggm ..... H51  
 \biggm ..... H48  
 \Biggr ..... H52  
 \biggr ..... H49  
 \Bigl ..... H44  
 \bigl ..... H41  
 \Bigm ..... H45  
 \bigm ..... H42  
 \bigodot ..... A353  
 \bigoplus ..... A352  
 \bigotimes ..... A351  
 \Bigr ..... H46  
 \bigr ..... H43  
 \bigskip ..... b454, o400  
 \bigskipamount ..... b453, o402, o403, O391  
 \bigsqcup ..... A356  
 \bigtriangledown ..... A361, A362  
 \bigtriangleup ..... A360, A363  
 \bigplus ..... A344  
 \bigvee ..... A342  
 \bigwedge ..... A343  
 \binoppenalty ..... b323  
 \bmod ..... H35  
 \boldmath ..... p14, z613  
 bool commands:  
 \bool\_gset\_false:N h15, U15, U81, U90  
 \bool\_gset\_true:N h10, U10, U121, U342

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

\bool\_if:NTF . . . . .  
     . . . . . 173, 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, U21, U88, U363  
\bool\_lazy\_and:nnTF . . . . . g653,  
     h132, h1006, h1229, U66, U112, U365  
\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, U6, U186, U201  
\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  
\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  
\bool\_while\_do:nn . . . . . h800  
\c\_false\_bool . . . . . 173, 258, g1948,  
     g2205, g2503, g2608, i140, i150, i288  
\c\_true\_bool . . . . . 173,  
     258, g1949, g2207, g2504, g2606, i145  
\BooleanFalse . . . . . g1596, g1941, g2503  
\BooleanTrue . . . . . g1595, g1940, g2503  
\bordermatrix . . . . . H172  
\bot . . . . . A321  
\botfigrule . . . . . V742, V2359  
\botmark . . . . . R50, V647, V706  
\bottomfraction . . . . . O275, V2328  
\bowtie . . . . . A482  
\Box . . . . . z726  
\box . . . . . 291, 645  
box commands:  
     \box\_dp:N . . . . . U192  
     \box\_gclear:N . . . . . n47  
     \box\_gset\_to\_last:N . . . . . n14, n75  
     \box\_ht:N . . . . . 868, U191, U300  
     \box\_if\_empty:NTF . . . . . U78, U97  
     \box\_if\_horizontal:NTF . . . . U232, U284  
     \box\_if\_vertical:NTF . . . . U206, U257  
     \box\_move\_up:nn . . . . . U246, U300  
     \box\_new:N . . . . . n42, U23, U25, U185, U202  
\box\_set\_dp:Nn . . . . . U219,  
     U228, U245, U270, U281, U299, U330  
\box\_set\_eq:NN . . . . . U148, U177, U182  
\box\_set\_eq\_drop:NN . . . . . U93  
\box\_set\_ht:Nn . . . . . U218,  
     U227, U244, U269, U280, U298, U329  
\box\_set\_wd:Nn . . . . . U217, U243, U268, U297  
\box\_use:N . . . . . U125, U223, U248, U276, U301, U331  
\box\_use\_drop:N . . . . . n44  
\box\_wd:N . . . . . U193, U294, U302  
\boxmaxdepth . . . . . b378,  
     L475, L503, L533, L611, L628,  
     V486, V506, V546, V715, V724, V764  
\brace . . . . . H59  
\braceld . . . . . A553, A557, A558, A560, A562  
\bracelu . . . . . A555, A559, A561  
\bracerd . . . . . A554, A559, A561  
\braceru . . . . . A556, A558, A562  
\bracevert . . . . . A608  
\brack . . . . . H58  
\break . . . . . b442, b447, f775, f796, o115  
\breve . . . . . A521  
\brokenpenalty . . . . . b328, u666  
\buildrel . . . . . A469, H149  
\bullet . . . . . A381

## C

\c . . . r234, r335, r337, r339, r341, r343,  
     r345, r347, r349, r351, r372, r374,  
     r392, r459, r478, r606, r608, r633,  
     r635, r648, r674, r701, r704, r705,  
     r706, r707, r708, r709, r710, r711,  
     r712, r762, r1242, r1256, r1282,  
     r1339, r1340, r1359, r1360, r1363,  
     r1364, r1369, r1370, r1381, r1382,  
     r1389, r1390, r1393, r1394, D167, D196  
\cal . . . . . z776  
call commands:  
     call\_callback . . . . . d688  
\call\_callback . . . . . 45  
callback commands:  
     callback\_descriptions . . . . . d818  
callback.register . . . . . d568  
\callback\_descriptions . . . . . 45  
\cap . . . . . A372  
\capitalacute . . . . .  
     r839, D160, D161, D193, D690, D943  
\capitalbreve . . . . .  
     r846, D162, D163, D194, D691, D950  
\capitalcaron . . . . .  
     r845, D164, D165, D195, D692, D949  
\capitalcedilla . . . . .  
     r832, D166, D167, D196, D693, D940

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
X=ltfinal.dtx

\capitalcircumflex ..... r840, D168, D169, D197, D694, D944  
 \capitaldieresis ..... r842, D170, D171, D198, D695, D946  
 \capitaldotaccent ..... r848, D172, D173, D199, D696, D952  
 \capitalgrave ..... r838, D174, D175, D200, D697, D942  
 \capitalhungarumlaut ..... r843, D176, D177, D201, D698, D947  
 \capitalmacron ..... r847, D178, D179, D202, D699, D951  
 \capitalnewtie ..... 576, r852, D190, D191, D203, D700, D1017, D1018  
 \capitalogonek ..... r835, D180, D181, D204, D701, D941  
 \capitalring ..... r844, D182, D183, D205, D702, D948  
 \capitaltie ..... r850, D184, D185, D206, D703, D1013, D1014  
 \capitaltilde ..... r841, D186, D187, D207, D704, D945  
 \caption ..... O4  
 \cases ..... H153, H154, H164, H166  
 \catcode ..... 613  
 \catcodetable ..... 41, d89, d109  
 \catcodetable@atletter ..... 42  
 \catcodetable@initex ..... 42  
 \catcodetable@latex ..... 42  
 \catcodetable@string ..... 42  
 \cdot ..... A394  
 \cdotp ..... A502, A508  
 \cdots ..... A508  
 center (environment) ..... G351  
 \center ..... G351  
 \centering ..... G351, G356, G357, G375, G377, G392, G394  
 \centerline ..... J489  
 \changes ..... 71, z667  
 \char ..... f737, f752, r391, r394, r430, r433, r444, r451, r477, r481, r486, r489, r491, r493, r733, r761, r764, r797, r804, r811, r834, r837, r866, r896, r1006, r1042, r1157, r1159, r1161, r1208, z626, z633, D631, D638, D640, D760, G466, G581, H238, L232, L282, L296, L304, L307, L448, L563, L568, L576, L580, L616, L617, L619, L632, L633, L636, L663  
 char commands:  
   \char\_generate:nn . e141, 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 ..... a64, a70, a71, b10, b16, b17, b18, 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, b463, b464, b465, d22, d26, d38, d47, d48, d89, d157, d216, j2, q52, q121, r18, u14, K4, K9, S1128, S1259, S1376, X28, X30, X34, X53, X157, X158, X159, X160, X161, X162, X163  
 \charsubdef ..... X339  
 \charzero ..... d216  
 \check ..... A522  
 \CheckCommand ..... f187  
 \CheckEncodingSubset . 570, D16, D61, D109, D110, D111, D156, D158, D352, D621, D832, D882, D938, D939, D1007, D1124, D1127, D1141  
 \chi ..... A288  
 \choose ..... H57  
 \circ ..... A391  
 \circle ..... L450, L605, L807, L824  
 \citation ..... Q11, Q19, Q47, Q64  
 \cite ..... 776  
 \cite ..... Q12  
 \clap ..... J493  
 class/.../after ..... 837  
 class/.../before ..... 837  
 class/after ..... 837  
 class/before ..... 837  
 \ClassError ..... 184  
 \ClassInfo ..... 184  
 \ClassNote ..... 1136  
 \ClassNoteNoLine ..... 1136  
 \ClassWarning ..... 184  
 \ClassWarningNoLine ..... 184  
 \cleaders ..... b483, A548, A551  
 \cleardoublepage ..... V138  
 \ClearHookNext ..... 180, h1414  
 \ClearHookRule ..... 184, h1451, h1554  
 \clearpage ..... 197, 334, 837, 838, 878, q296, q323, q327, q351, q369, G17, G72, G165, V125

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

V138, V143, V200, V407, V410,	g747
V414, V455, V461, V2205, V2222	g755
\cline ..... . . . . . K367	g779
clist commands:	g787
\clist_gclear:N ..... h799	g795
\clist_gput_left:Nn ..... h745	g803
\clist_gput_right:Nn ..... h747	g593
\clist_if_empty:NTF ..... h1023	g501, g513, g534, g555, g593
\clist_map_inline:nn ..... h572, h580	g103, 117
\clist_new:N ..... h86	103, 117
\clist_use:Nn ..... h1025	g11, g88, g379, g466, g535, g668
\clubpenalty ..... 297, 326, b325,	g259, g265, g270, g271, g273
q8, q25, q94, q151, u664, N100,	103, 117
N106, N130, N135, I128, I194, I196	g14, g269, g271, g273, g311, g316, g323
\clubsuit ..... A331	124, g249, g309
cmd internal commands:	g309
\__cmd_add_arg:n ..... . . . . .	g309
. 141, g1361, g1372, g1377, g1378,	103, 117
g1412, g1500, g1507, g1582, g1595,	g14, g269, g271, g273, g311, g316, g323
g1596, g1669, g1727, g1734, g1747	124, g249, g309
\__cmd_add_arg_spec:n ..... . . . . .	g309
. . . . . g503, g517, g577, g651	103, 117
\__cmd_add_arg_spec_mandatory:n .. . . . .	g549, g559, g565, g651
\__cmd_add_default: ..... . . . . .	103, 117
. 124, g743, g781, g798, g844, g931, g940	103, 117
\__cmd_add_default:n ..... . . . . .	124, g750, g790, g806, g844, g875
\__cmd_add_default_E:nn ..... . . . . .	124, g750, g790, g806, g844, g875
\__cmd_add_expandable_grabber:nn .. . . . .	124, g750, g790, g806, g844, g875
\__cmd_add_expandable_type_+:w ..... g866	124, g750, g790, g806, g844, g875
\__cmd_add_expandable_type_D:w ..... g871	124, g750, g790, g806, g844, g875
\__cmd_add_expandable_type_D_- aux:NN ..... g871	124, g750, g790, g806, g844, g875
\__cmd_add_expandable_type_D_- aux:NNN ..... g871	124, g750, g790, g806, g844, g875
\__cmd_add_expandable_type_D_- aux:NNNn ..... g871, g937	124, g750, g790, g806, g844, g875
\__cmd_add_expandable_type_E:w ..... g907	124, g750, g790, g806, g844, g875
\__cmd_add_expandable_type_E_- aux:n ..... g907	124, g750, g790, g806, g844, g875
\__cmd_add_expandable_type_m:w ..... g929	124, g750, g790, g806, g844, g875
\__cmd_add_expandable_type_R:w ..... g936	124, g750, g790, g806, g844, g875
\__cmd_add_expandable_type_t:w ..... g938	124, g750, g790, g806, g844, g875
\__cmd_add_grabber:N ..... . . . . .	124, g750, g790, g806, g844, g875
. 126, g744, g751, g764, g783, g791, g799, g807, g821	124, g750, g790, g806, g844, g875
\__cmd_add_type_!:_w ..... g725	124, g750, g790, g806, g844, g875
\__cmd_add_type_+_w ..... g718	124, g750, g790, g806, g844, g875
\__cmd_add_type_>:_w ..... g732	124, g750, g790, g806, g844, g875
\__cmd_add_type_b:w ..... g740	124, g750, g790, g806, g844, g875
\__cmd_add_type_D:w ..... . . . . .	124, g750, g790, g806, g844, g875
\__cmd_add_type_E:w ..... . . . . .	124, g750, g790, g806, g844, g875
\__cmd_add_type_m:w ..... . . . . .	124, g750, g790, g806, g844, g875
\__cmd_add_type_R:w ..... . . . . .	124, g750, g790, g806, g844, g875
\__cmd_add_type_t:w ..... . . . . .	124, g750, g790, g806, g844, g875
\__cmd_allowed_token_check:N ..... . . . . .	124, g750, g790, g806, g844, g875
\__cmd_arg_spec_t1 ..... 103, 117	124, g750, g790, g806, g844, g875
\__cmd_args_i_t1 ..... . . . . .	124, g750, g790, g806, g844, g875
\__cmd_args_ii_t1 ..... 103, 117	124, g750, g790, g806, g844, g875
\__cmd_args_process: ..... 124, g249, g309	124, g750, g790, g806, g844, g875
\__cmd_args_process_aux:n ..... g309	124, g750, g790, g806, g844, g875
\__cmd_args_process_loop:nn .. g309	124, g750, g790, g806, g844, g875
\__cmd_args_t1 ..... . . . . .	124, g750, g790, g806, g844, g875
. 103, 106, 113, 141, 146, g12, g212, g237, g254, g259, g265, g284, g313, g316, g329, g330, g1321, g1322, g1327, g1330, g1478, g1512, g1749	124, g750, g790, g806, g844, g875
\__cmd_bad_arg_spec:wn ..... g436, g441, g450, g459, g473, g486, g500, g510, g511, g516, g527, g533, g554, g645	124, g750, g790, g806, g844, g875
\__cmd_bad_def:wn ..... . . . . .	124, g750, g790, g806, g844, g875
. 132, g392, g400, g429, g464, g478, g491, g574, g582, g590, g609, g618, g630, g645, g661, g683	124, g750, g790, g806, g844, g875
\__cmd_bool_reverse:N ..... g1945, g2626	124, g750, g790, g806, g844, g875
\__cmd_break_point:n ..... g91, g93, g645, g650, g998, g1005, g1176	124, g750, g790, g806, g844, g875
\__cmd_cant_copy:nwn ..... . . . . .	124, g750, g790, g806, g844, g875
. g995, g1005, g1082, g1145, g1174	124, g750, g790, g806, g844, g875
\__cmd_check_definable:nNTF ..... g2093, g2507, g2520, g2533, g2538, g2563, g2576, g2589, g2597, g2633, g2639	124, g750, g790, g806, g844, g875
\__cmd_check_definable_aux:n ..... . . . . .	124, g750, g790, g806, g844, g875
. 163, g2093	124, g750, g790, g806, g844, g875
\__cmd_check_end:n ..... g1141	124, g750, g790, g806, g844, g875
\__cmd_check_end:Nn ..... . . . . .	124, g750, g790, g806, g844, g875
. g1128, g1129, g1141, g1220	124, g750, g790, g806, g844, g875
\__cmd_check_end:w ..... g1141	124, g750, g790, g806, g844, g875
\__cmd_chk_if_free_cs:N ..... . . . . .	124, g750, g790, g806, g844, g875
. g2065, g2643, g2644	124, g750, g790, g806, g844, g875
\__cmd_cmd_if_xparse_aux:N ..... g2160	124, g750, g790, g806, g844, g875
\__cmd_cmd_type_cases:Nnnnn ..... g2160	124, g750, g790, g806, g844, g875
\__cmd_cmd_type_cases:NnnnnTF ..... . . . . .	124, g750, g790, g806, g844, g875
. 132, 137, g990, g1169, g2160, g2190	124, g750, g790, g806, g844, g875
\__cmd_copy:NN ..... 133, g983, g985, g1159	124, g750, g790, g806, g844, g875
\__cmd_copy_command:nnNN ..... . . . . .	124, g750, g790, g806, g844, g875
. 133, g991, g1020	124, g750, g790, g806, g844, g875

**File Key:** a=ltirchck.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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

```

__cmd_copy_command:NnNNnnnn . . .
__cmd_copy_environment:nnNN . . .
__cmd_copy_environment:Nnnnnnn . . .
__cmd_copy_environment_end:nnNN . . .
__cmd_copy_environment_end_- aux:nnNN . . .
__cmd_copy_expandable:nnN . . .
__cmd_copy_expandable:nnNN . . .
__cmd_copy_expandable:NnNNNNnnn . . .
__cmd_copy_expandable_signature:NnNNNNnnn . . .
__cmd_copy_grabber_{type}:w . . .
__cmd_copy_grabber_D:w . . .
__cmd_copy_grabber_D_alt:w . . .
__cmd_copy_grabber_E:w . . .
__cmd_copy_grabber_E_long:w . . .
__cmd_copy_grabber_m:w . . .
__cmd_copy_grabber_m_long:w . . .
__cmd_copy_grabber_R:w . . .
__cmd_copy_grabber_R_alt:w . . .
__cmd_copy_grabber_t:w . . .
__cmd_copy_parse_grabber:w . . .
\l__cmd_current_arg_int . . .
__cmd_declare_cmd:Nnn . . .
__cmd_declare_cmd_aux:Nnn . . .
__cmd_declare_cmd_code:Nnn . . .
__cmd_declare_cmd_code_aux:Nnn . . .
__cmd_declare_cmd_code_expandable:Nnn . . .
__cmd_declare_cmd_internal:Nnnn . . .
__cmd_declare_env:nnnn . . .
__cmd_declare_env_internal:nnnn . . .
__cmd_declare_expandable_- cmd:Nnn . . .
__cmd_defaults: g248, g256
__cmd_defaults_aux: g256
\l__cmd_defaults_bool . . .
__cmd_defaults_def: g256
__cmd_defaults_def:nn g256
__cmd_defaults_def:nnn g256
__cmd_defaults_error:w g256
\l__cmd_defaults_t1 . . .
__cmd_end_expandable>NNw g336, g338
__cmd_end_expandable_aux:nNNNN g338
__cmd_end_expandable_aux:w . . g338
__cmd_end_expandable_defaults:nnnNn . . .
__cmd_end_expandable_defaults:nnNn . . .
__cmd_end_expandable_defaults:nw . . .
\l__cmd_environment_bool . . .
__cmd_end_expandable_defaults:nnw . . .
__cmd_end_expandable_defaults:nw . . .
\l__cmd_environment_or_command: . . .
__cmd_environment_str . . .
__cmd_expandable_aux_name_t1 . . .
__cmd_expandable_bool . . .
\l__cmd_expandable_grab_D:nnNNNwNN . . .
__cmd_expandable_grab_D:NNNwNNn . . .
__cmd_expandable_grab_D:NNNwNNnn . . .
__cmd_expandable_grab_D:Nw . . .
__cmd_expandable_grab_D:w . . .
__cmd_expandable_grab_D_- alt:NNwn . . .
__cmd_expandable_grab_D_- alt:NNwNNn . . .
__cmd_expandable_grab_D_alt:Nwn . . .

```

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

```

__cmd_expandable_grab_D_alt:w g1805
__cmd_expandable_grab_E:w ... g1840
__cmd_expandable_grab_E_aux:w g1840
__cmd_expandable_grab_E_end:nnw
 g1840
__cmd_expandable_grab_E_-
 find:nnw g1840
__cmd_expandable_grab_E_find:w .
 g1840
__cmd_expandable_grab_E_long:w .
 g1840
__cmd_expandable_grab_E_-
 loop:nnnNNw g1840
__cmd_expandable_grab_E_-
 test:nnw g1840
__cmd_expandable_grab_m:w 135, g1875
__cmd_expandable_grab_m_aux:wNn
 g1875
__cmd_expandable_grab_m_long:w .
 g1875
__cmd_expandable_grab_R:w ... g1881
__cmd_expandable_grab_R_alt:w g1908
__cmd_expandable_grab_R_alt_-
 aux:NNwNNn g1908
__cmd_expandable_grab_R_-
 aux:NNNwNNn g1881
__cmd_expandable_grab_t:w ... g1935
__cmd_expandable_grab_t_-
 aux:NNwn g1935
__cmd_flush_m_args: 124,
 126, g701, g720, g727, g734, g742,
 g749, g757, g789, g797, g805, g810
\l__cmd_fn_code_t1 104, g29, g239, g254
\l__cmd_fn_t1
 ... 104, 107, 141, 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 105, 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, g2634
__cmd_get_arg_spec:n . g2068, g2636
__cmd_get_arg_spec:NTF
 ... g2056, g2070, g2075, g2082, g2088
__cmd_get_arg_spec_error:N
 g2031, g2071, g2084
__cmd_get_arg_spec_error:n
 g2031, g2078, g2091
__cmd_get_arg_spec_error_aux:n .
 g2031
__cmd_get_grabber:NN
 130, g925, g941, g954
__cmd_get_grabber_auxi:NN ... g954
__cmd_get_grabber_auxii:NN .. g954
__cmd_grab_b:w g1309
__cmd_grab_b_aux:NNw g1309
__cmd_grab_b_end:Nw g1309
__cmd_grab_b_long:w g1309
__cmd_grab_b_long_obey_spaces:w
 g1309
__cmd_grab_b_obey_spaces:w ... g1309
__cmd_grab_D:w g1336
__cmd_grab_D_aux:NNnN
 g1319, g1356, g1575
__cmd_grab_D_aux:NNnNN
 ... g1338, g1343, g1348, g1353, g1356
__cmd_grab_D_call:Nw
 142, g1360, g1420, g1577
__cmd_grab_D_long:w g1336
__cmd_grab_D_long_obey_spaces:w
 g1336
__cmd_grab_D_nested:NNnN
 g1369, g1383
__cmd_grab_D_nested:w g1383
__cmd_grab_D_obey_spaces:w ... g1336
__cmd_grab_E:nnNN
 g1439
__cmd_grab_E:w g1439
__cmd_grab_E_finalise: g1439
__cmd_grab_E_long:w g1439
__cmd_grab_E_long_obey_spaces:w
 g1439
__cmd_grab_E_loop:NnN g1439
__cmd_grab_E_obey_spaces:w ... g1439
\l__cmd_grab_expandably_bool ...
 105, 115, g31,
 g96, g380, g404, g406, g468, g505,
 g519, g541, g558, g620, g663, g714
__cmd_grab_m:w 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

```

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
X=ltfinal.dtx

```

__cmd_grab_m_aux:Nnnnnnnnn . . g1510
__cmd_grab_m_long:w g1496
__cmd_grab_R:w g1569
__cmd_grab_R_aux:NNnN g1569
__cmd_grab_R_long:w g1569
__cmd_grab_t:w g1585
__cmd_grab_t_aux>NNw g1585
__cmd_grab_t_obey_spaces:w . . g1585
__cmd_grab_v:w g1601
__cmd_grab_v_aux:w 150, g1601
__cmd_grab_v_aux_abort:n
. 149, g1627, g1645,
g1651, g1664, g1683, g1706, g1708
__cmd_grab_v_aux_catcodes: . . .
. 148, g1642, g1674, g1708
__cmd_grab_v_aux_loop:N g1637
__cmd_grab_v_aux_loop:NN g1637
__cmd_grab_v_aux_loop_end: . . .
. g1637, g1697
__cmd_grab_v_aux_put:N g1641,
g1660, g1676, g1694, g1702, g1737
__cmd_grab_v_aux_test:N g1626, g1637
__cmd_grab_v_bgroup: . . g1622, g1672
__cmd_grab_v_bgroup_loop: . . g1672
__cmd_grab_v_bgroup_loop:N . . g1672
__cmd_grab_v_group_end: . . .
. 149, g1601, g1668, g1719
__cmd_grab_v_long:w g1601
__cmd_grab_v_token_if_char:NTF .
. 149, g1639, g1655, g1687, g1745
\g__cmd_grabber_int . . .
. 104, g27, g967, g971
\c__cmd_ignore_def_tl 166, g2240,
g2258, g2265, g2279, g2300, g2329,
g2336, g2343, g2351, g2359, g2368,
g2375, g2382, g2389, g2401, g2408
\l__cmd_last_delimiters_tl 105,
115, 118, g33, g378, g396, g504,
g518, g540, g576, g626, g636, g685
\l__cmd_long_bool . . .
. 105, 129, 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 105,
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
__cmd_normalize_arg_spec_loop:n
. g375, g469, g482, g496,
g506, g521, g544, g550, g560, g566
__cmd_normalize_check_gv:N . . .
. g564, g612
__cmd_normalize_check_lu:N . . g612
__cmd_normalize_E_unique_-
check:w g498
__cmd_normalize_type_!w g457
__cmd_normalize_type_+w g457
__cmd_normalize_type_>w g457
__cmd_normalize_type_b:w g568
__cmd_normalize_type_D:w . . .
. g437, g445, g447, g498
__cmd_normalize_type_d:w g432
__cmd_normalize_type_E:w g442, g498
__cmd_normalize_type_e:w g432
__cmd_normalize_type_m:w g546
__cmd_normalize_type_0:w g432
__cmd_normalize_type_o:w g432
__cmd_normalize_type_R:w g451, g546
__cmd_normalize_type_r:w g432
__cmd_normalize_type_s:w g432
__cmd_normalize_type_t:w g454, g498
__cmd_normalize_type_v:w g546
\l__cmd_obey_spaces_bool 105, 127,
g32, g381, g487, g493, g537, g542,
g671, g675, g692, g728, g829, g834
__cmd_peek_cs_check_equal:NNN g2204
__cmd_peek_meaning:NTF g2195, g2204
__cmd_peek_meaning_aux:NNTF . g2204
__cmd_peek_meaning_remove:NTF .
. g1349, g1354,
g1455, g1461, g1588, g2197, g2204
__cmd_peek_nonspace:NTF g2194
__cmd_peek_nonspace_aux:nNNTF g2194
__cmd_peek_nonspace_remove:NTF .
. g1339, g1344,
g1443, g1449, g1576, g1586, g2194
__cmd_peek_true_remove:NNw . . g2204
__cmd_peek_true_remove:Nw . . .
. g2215, g2219, g2227, g2231
\l__cmd_prefixed_bool . . .
. 105, g36, g705, g722, g729, g735, g782
__cmd_prepare_signature:N . . .
. 124, g688,
g745, g753, g766, g785, g793, g801,
g808, g869, g879, g921, g934, g947
__cmd_prepare_signature:n g88, g688
__cmd_prepare_signature_-
bypass:N 124, g688, g723, g730, g738

```

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
X=ltfinal.dtx

```

\l__cmd_process_all_tl 105, 113, g37, g124,
g241, g249, g314, g696, g816, g835
\l__cmd_process_one_tl . 105, 126,
g38, g697, g737, g762, g771, g839, g842
\l__cmd_process_some_bool 105, g39, g123, g698, g736
\l__cmd_put_arg_expandable:nw ...
..... g1789, g1794,
g1795, g1830, g1835, g1836, g1943
\l__cmd_replicate_processor:nn ...
..... g761, g768
\l__cmd_run_code: 111,
g243, 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 106, g40, g1321, g1329
\l__cmd_set_environment_end:n ...
..... g207, g251
\l__cmd_set_eq_if_exist>NN . g985,
g1023, g1036, g1037, g1038, g1118
\l__cmd_show:N
.... 138, g1162, g1164, g1224, g1306
\l__cmd_show_arg_spec:N . g2080, g2640
\l__cmd_show_arg_spec:n . g2080, g2642
\l__cmd_show_command:N . g1170, g1180
\l__cmd_show_command:NnNNwN ... g1180
\l__cmd_show_command_aux:nNNn ...
..... 137, 138, g1180
\l__cmd_show_delim:Nw g1254
\l__cmd_show_delims:Nw g1254
\l__cmd_show_delims_opt:Nw ... g1254
\l__cmd_show_E:Nw g1254
\l__cmd_show_e:Nw g1254
\l__cmd_show_environment:N ...
..... g1172, g1180
\l__cmd_show_environment:Nnnw ...
..... g1200, g1208
\l__cmd_show_environment_end:N ...
..... g1173, g1218
\l__cmd_show_expandable:N g1171, g1180
\l__cmd_show_expandable:NnNNNNn ...
..... g1180
\l__cmd_show_opt:Nw g1254
\l__cmd_show_prefix:Nw g1254
\l__cmd_show_processor:Nw ... g1254
\c__cmd_show_type_!_tl g1248
\c__cmd_show_type_+_tl g1248
\c__cmd_show_type_>_tl g1248
\c__cmd_show_type_D_t1 g1248
\c__cmd_show_type_d_t1 g1248
\c__cmd_show_type_E_t1 g1248
\c__cmd_show_type_e_t1 g1248
\c__cmd_show_type_0_t1 g1248
\c__cmd_show_type_R_t1 g1248
\c__cmd_show_type_r_t1 g1248
\c__cmd_show_type_t_t1 g1248
\l__cmd_signature_t1
... 106, 141, 152, 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
\l__cmd_single_token_check:n g501,
g502, g512, g534, g555, g556, g584
\l__cmd_some_long_bool 106,
115, g43, g154, g162, g384, g655, g666
\l__cmd_some_obey_spaces_bool ...
..... 106, g42, g383, g494, g679
\l__cmd_some_short_bool
.... 106, 115, g44, g153, g164, g385, g667
\l__cmd_split_add_item:n ...
..... 140, g1287, g1289, g1290
\l__cmd_split_argument:nnm
..... g1986, g2627
\l__cmd_split_argument_aux:n .. g1986
\l__cmd_split_argument_aux:nnnn g1986
\l__cmd_split_argument_aux:wn .. g1986
\l__cmd_split_end_item:n ...
..... 140, g1246, g1255, g1257,
g1259, g1261, g1267, g1278, g1290
\l__cmd_split_list:nn ...
..... g1951, g1988, g2628
\l__cmd_split_list_multi:nn ... g1951
\l__cmd_split_list_seq
.... 159, g1951, g1965, g1967
\l__cmd_split_list_single:Nn .. g1951
\l__cmd_split_list_tl
.... 159, g1952, g1975, g1981, g1983
\l__cmd_split_signature:n
.... 137, g1190, g1225
\l__cmd_split_signature_loop:Nw ...
..... g1230, g1232
\l__cmd_split_start_item:
139, 140, g1235, g1266, g1277, g1290
\l__cmd_start:nNNnnn
.... 108, 125, 164, g113, g215, g2176
\l__cmd_start_aux:NNnnnn
..... g221, g232, g235

```

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
X=ltfinal.dtx

\__cmd_start_env:nnnn . . . . .	H18
. . . . . 164, g109, g215, g2178	
\__cmd_start_expandable:nNNNNn . .	H19
. . . . . 108, 164, g147, g333, g2177	
\__cmd_t1_mapthread_function:NNN . . . . .	X61
. . . . . g284, g312, g2134	
\__cmd_t1_mapthread_function:nnN . . . . .	
. . . . . 140, g360, g1280, g2134	
\__cmd_t1_mapthread_loop:w . . . . .	
. . . . . g2134	
\__cmd_tmp:w . . . . .	d217
. . . . . 112, 117, 153, g45, 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 . . . . .	
. . . . . 106, g45, g1467, g1470, g1476	
\l__cmd_tmpa_t1 . . . . .	
106, 137, 138, 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_tmrb_t1 . . . . .	
106, 130, 138, 139, g47, g910, g915, g926, g1129, g1131, g1229, g1235, g1292, g1296, g1300, g1301, g1476, g1477, g1479, g2211, g2222, g2228	
\__cmd_token_if_cs:NTF . . . . .	
. . . . . 120, g1428, g2125, g2217	
\__cmd_trim_spaces:n . . . . .	
. . . . . g2029, g2629	
\l__cmd_v_arg_t1 . . . . .	
. . . . . 148, 149, g1600, g1616, g1635, g1669, g1725, g1732, g1739	
\l__cmd_v_nesting_int . . . . .	
150, g1671, g1675, g1691, g1692, g1701	
\colon . . . . .	A503
\columnsep . . . . .	V81, V202
\columnseprule . . . . .	V82, V2260, V2294
\columnwidth . . . . .	
q24, q27, q28, q30, q93, q96, q97, q99, q150, q153, q154, q157, J346, J377, J395, J412, O99, O168, O470, O489, O507, V80, V144, V145, V146, V201, V202, V203, V204, V205, V1843, V1845, V2258, V2262, V2290, V2296	
\cong . . . . .	A457
\contentsline . . . . .	
. . . . . 746, 747, N164, N171, N177, N184	
\coprod . . . . .	A341
\copyright . . . . .	r285, r316, z649
\cos . . . . .	H12
\cosh . . . . .	H14
\cot . . . . .	H18
\coth . . . . .	H19
\countdef . . . . .	
a66, b37, b38, b39, b41, b51, b90, d75, d85, d174, d182, d190, d198, d217, E3, X61	
\counterwithin . . . . .	
391	
\counterwithin . . . . .	
s75	
\counterwithout . . . . .	
391	
\counterwithout . . . . .	
s75	
\CountZero . . . . .	
\cr . . . . .	
b413, r500, r506, r516, r522, D101, D105, D930, D934, H175, H179, H365, H411, H510, K192, K203, K210, K219, K224, K230, K377, L141, L143, L148, L154	
\crcr . . . . .	
b470, r327, r362, r363, r390, r394, r397, r476, r480, r484, r486, r489, r732, r760, r764, r767, r834, r837, r895, r1245, z651, A337, A338, A340, A459, A462, A466, A530, A531, A532, A533, A534, A535, A537, A538, A539, A540, A541, A543, D106, D935, H155, H157, H158, H159, H175, H177, H178, H179, H197, H198, K171, K172, L141	
create commands:	
create_callback . . . . .	d663
\create_callback . . . . .	45
\CS . . . . .	75
\cs . . . . .	h1421, h1422, h1424
cs commands:	
\cs:w . . . . .	
209, g1431, g2181, h219, h704, h738, h739, h791, h811, h814, h830, h831, h854, h855, h856, h863, h870, h1122, h1128, h1141, h1151, h1187, h1217, h1262, h1292, y187, y190, T211, T225, T237, T238	
\cs_argument_spec:N . . . . .	
. . . . . 259, 264, e138, 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, y187, y190, T211, T225, T237, T238	
\cs_generate_from_arg_count:NNnn . . . . .	
. . . . . 102, 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, T428, T436	
\cs_gset:Npn . . . . .	U171

**File Key:** a=ltirchck.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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

```

\cs_gset:Npx T213
\cs_gset_eq:NN e135, e136, e137, e138,
e139, e140, e141, g2066, g2643,
h451, h711, h724, h725, h1114, i55,
i384, y184, y198, U52, U150, U164,
U165, U170, U179, U183, U319, U376
\cs_gset_nopar:Npx h47, h49, h51
\cs_gset_protected:Npn h1434
\cs_gset_protected:Npx h20, U20
\cs_if_eq:NNTF g962, g1487
\cs_if_exist:NTF 852, g63,
g179, g965, g1003, g2035, g2042,
g2509, g2522, g2534, g2543, g2546,
g2553, g2558, g2565, g2578, g2591,
h1272, h1280, i72, i80, T93, T405, T409
\cs_if_exist_p:N g70, g71
\cs_if_exist_use:NTF g412,
g1081, h652, U311, U312, U316, U317
\cs_new:Npn 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, T30, T38, T44,
T57, T95, T100, T105, T112, T127,
T234, T240, T378, T380, T382,
T385, T389, T403, T419, T429,
T545, U54, U59, U77, U140, U145,
U161, U176, U181, U187, U190,
U204, U255, U308, U321, U335,
U338, U348, U381, U500, U501, U502
\cs_new_eq:NN g48,
g93, g1099, g1100, g1107, g1114,
g2065, g2503, g2504, g2619, g2620,
g2621, g2626, g2627, g2628, g2629,
g2630, g2636, g2642, h7, h23, h34,
h64, h665, h671, h709, h909, h910,
h911, h912, h913, h914, h1160,
h1453, h1454, h1524, h1526, i12,
i13, i335, n93, T252, T253, T472,
T474, T476, T478, T480, T482,
T484, T486, T488, T490, U7, U151,
U345, U347, U400, U403, U404, U499
\cs_new_protected:Npn 880, 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,
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, g2631, g2637, 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,

```

**File Key:** a=ltirchck.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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
X=ltfinal.dtx

\cs\_to\_str:N ..... 105, 209, e135, g70, g71, g84, g996, g998, g1176, g1203, g1205, g1220, g1431, g2183, i111, i132, i146, i151

\cs\_undefine:N ..... g2644, h128, h685, i58, T227

cs\check@icr commands:

- \cs\_gset\_eq:NN ..... 71
- \csc ..... H21
- \csname ..... 181, 515, 617, 848, 851
- \csname\endcsname ..... 848, 850
- \cup ..... A373
- \CurrentFile ..... 334, 812, 836, 839, 842, 844, 845, 850, S854, T3, T67, T84, T166, T171, T174, T344, T348, T518, T519
- \CurrentFilePath ..... 836, 842, 845, 850, T3, T67, T83, T165, T345, T348
- \CurrentFilePathUsed ..... 836, T3, T66, T81, T163, T342, T345
- \CurrentFileUsed ..... 812, 836, 839, 844, 850, S854, T3, T66, T82, T164, T342, T344, T517, T518
- \CurrentOption ..... r1528, r1531, r1536, r1548, S14, S455, S466, S479, S480, S485, S498, S499, S501, S515, S516, S519, S533, S538, S539, S552, S557, S558, S571, S573, S579, S581, S593, S594, S595, S603, S604, S605, S887, S952, S1050, S1051, S1061

CurrentOption commands:

- \CurrentOption: ..... S478, S497, S514, S532, S537, S551, S556, S592, S602, S1060
- \CYRA ..... r1500
- \cyra ..... r1500, r1551
- \CYRABHCH ..... r1500
- \cyrabhch ..... r1500
- \CYRABHCHDSC ..... r1500
- \cyrabhchdsc ..... r1500
- \CYRABHDZE ..... r1501
- \cyrabhdze ..... r1500
- \CYRABHHA ..... r1501
- \cyrabhha ..... r1501
- \CYRAE ..... r1501
- \cyrae ..... r1501
- \CYRB ..... r1501
- \cyrb ..... r1501
- \CYRBYUS ..... r1502
- \cyrbyus ..... r1501
- \CYRC ..... r1502
- \cyrc ..... r1502
- \CYRCH ..... r1502
- \cyrch ..... r1502

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfnccmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmiscren.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltlyphen.dtx, X=ltfinal.dtx

\CYRCHLDSC . . . . .	r1502	\CYRII . . . . .	r1509
\cyrchldsc . . . . .	r1502	\cyrii . . . . .	r1509
\CYRCHRDSC . . . . .	r1503	\CYRISHRT . . . . .	r1509
\cyrchrdsc . . . . .	r1502	\cyrishrt . . . . .	r1509
\CYRCHVCRS . . . . .	r1503	\CYRISHRTDSC . . . . .	r1510
\cyrchvcrs . . . . .	r1503	\cyrishrtdsc . . . . .	r1510
\CYRD . . . . .	r1503	\CYRIZH . . . . .	r1510
\cyrd . . . . .	r1503	\cyrizh . . . . .	r1510
\CYRDELTA . . . . .	r1503	\CYRJE . . . . .	r1510
\cyrdelta . . . . .	r1503	\cyrje . . . . .	r1510
\CYRDJE . . . . .	r1504	\CYRK . . . . .	r1510
\cyrdje . . . . .	r1504	\cyrk . . . . .	r1510
\CYRDZE . . . . .	r1504	\CYRKBEAK . . . . .	r1511
\cyrdze . . . . .	r1504	\cyrkbeak . . . . .	r1511
\CYRDZHE . . . . .	r1504	\CYRKDSC . . . . .	r1511
\cyrdzhe . . . . .	r1504	\cyrkdsc . . . . .	r1511
\CYRE . . . . .	r1504	\CYRKHCRS . . . . .	r1511
\cyre . . . . .	r1504	\cyrkhcrs . . . . .	r1511
\CYREPS . . . . .	r1505	\CYRKHK . . . . .	r1512
\cyreps . . . . .	r1504	\cyrkhk . . . . .	r1511
\CYREREV . . . . .	r1505	\CYRKVCRS . . . . .	r1512
\cyrerev . . . . .	r1505	\cyrkvcrs . . . . .	r1512
\CYRERY . . . . .	r1505	\CYRL . . . . .	r1512
\cyrery . . . . .	r1505	\cyrl . . . . .	r1512
\CYRF . . . . .	r1505	\CYRLDSC . . . . .	r1512
\cyrf . . . . .	r1505	\cyrldsc . . . . .	r1512
\CYRFITA . . . . .	r1506	\CYRLHK . . . . .	r1513
\cyrfita . . . . .	r1505	\cyrlhk . . . . .	r1512
\CYRG . . . . .	r1506	\CYRLJE . . . . .	r1513
\cyrg . . . . .	r1506	\cyrlje . . . . .	r1513
\CYRGDSC . . . . .	r1506	\CYRM . . . . .	r1513
\cyrgdsc . . . . .	r1506	\cyrm . . . . .	r1513
\CYRGDSCHCRS . . . . .	r1506	\CYRMDSC . . . . .	r1513
\cyrgdschcrs . . . . .	r1506	\cyrmdsc . . . . .	r1513
\CYRGHCRS . . . . .	r1507	\CYRMHK . . . . .	r1513
\cyrghcrs . . . . .	r1507	\cyrmhk . . . . .	r1513
\CYRGHK . . . . .	r1507	\CYRN . . . . .	r1514
\cyrghk . . . . .	r1507	\cyrn . . . . .	r1514
\CYRGUP . . . . .	r1507	\CYRNDSC . . . . .	r1514
\cyrgup . . . . .	r1507	\cyrndsc . . . . .	r1514
\CYRH . . . . .	r1507	\CYRNG . . . . .	r1514
\cyrh . . . . .	r1507	\cyrng . . . . .	r1514
\CYRHDESC . . . . .	r1508	\CYRNHK . . . . .	r1514
\cyrhdsc . . . . .	r1508	\cyrnhk . . . . .	r1514
\CYRHHCRS . . . . .	r1508	\CYRNJE . . . . .	r1515
\cyrhhcrs . . . . .	r1508	\cyrnje . . . . .	r1514
\CYRHHK . . . . .	r1508	\CYRNLHK . . . . .	r1515
\cyrhhk . . . . .	r1508	\cyrnlhk . . . . .	r1515
\CYRHRDSN . . . . .	r1509	\CYRO . . . . .	r1515
\cyrhrdsn . . . . .	r1508	\cyro . . . . .	r1515
\CYRI . . . . .	r1509	\CYROTLD . . . . .	r1515
\cyri . . . . .	r1509	\cyrotld . . . . .	r1515
\CYRIE . . . . .	r1509	\CYRP . . . . .	r1515
\cyrie . . . . .	r1509	\cyrp . . . . .	r1515

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lpictur.dtx,  
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
X=ltfinal.dtx

\CYRPHK . . . . .	r1516	\CYRYI . . . . .	r1522
\cyrphk . . . . .	r1516	\cyri . . . . .	r1522
\CYRQ . . . . .	r1516	\CYRYO . . . . .	r1523
\cyrq . . . . .	r1516	\cyro . . . . .	r1522
\CYRR . . . . .	r1516	\CYRYU . . . . .	r1523
\cyrr . . . . .	r1516	\cyru . . . . .	r1523
\CYRRDSC . . . . .	r1516	\CYRZ . . . . .	r1523
\cyrrdsc . . . . .	r1516	\cyrz . . . . .	r1523
\CYRRHK . . . . .	r1517	\CYRZDSC . . . . .	r1523
\cyrrhk . . . . .	r1516	\cyrzdsc . . . . .	r1523
\CYRTICK . . . . .	r1517	\CYRZH . . . . .	r1523
\cyrtick . . . . .	r1517	\cyrzh . . . . .	r1523
\CYRS . . . . .	r1517	\CYRZHDSC . . . . .	r1524
\cyrs . . . . .	r1517	\cyrzhdsc . . . . .	r1524
\CYRSACRS . . . . .	r1517		
\crysacrs . . . . .	r1517		
\CYRSCHWA . . . . .	r1518	<b>D</b>	
\cryschwa . . . . .	r1518	\d . . . . .	r235, r395,
\CYRSDSC . . . . .	r1518		r482, r765, r1241, r1455, r1456,
\crysdesc . . . . .	r1518		r1457, r1458, r1461, r1462, r1463,
\CYRSEMISFTSN . . . . .	r1518		r1464, r1465, r1466, r1467, r1468,
\cryssemisftsn . . . . .	r1518		r1469, r1470, r1471, r1472, r1473,
\CYRSFTSN . . . . .	r1519		r1474, r1475, r1476, r1477, r1478,
\crysftsn . . . . .	r1519		r1479, r1480, r1481, r1482, r1483,
\CYRSH . . . . .	r1519		r1484, r1485, r1486, r1487, r1488,
\cyrsh . . . . .	r1519		r1489, r1490, r1491, r1492, r1493, r1494
\CYRSHCH . . . . .	r1519	\dag . . . . .	r312
\cyrshch . . . . .	r1519	\dagger . . . . .	r312, s165, s171, s179, s180, A375
\CYRSHHA . . . . .	r1519	\dashbox . . . . .	<u>L309</u> , L808, L825
\cyrshha . . . . .	r1519	\dashv . . . . .	A403
\CYRT . . . . .	r1520	\date . . . . .	737
\cyrt . . . . .	r1520	\date . . . . .	N9, N25
\CYRTDSC . . . . .	r1520	\day . . . . .	a185, S1172, S1305, S1394
\cyrtdesc . . . . .	r1520	\dblfigrule . . . . .	V767, V2359
\CYRTETSE . . . . .	r1520	\dblfloatpagefraction . . . . .	O287, O301, V2336
\cyrtetse . . . . .	r1520	\dblfloatsep . . . . .	
\CYRTSHE . . . . .	r1520		V753, V765, V1640, V1766, V2343
\cyrtshe . . . . .	r1520	\dbltextfloatsep . . . . .	V222,
\CYRU . . . . .	r1521		V230, V769, V1639, V1765, V2343
\cyru . . . . .	r1521	\dbltopfraction . . . . .	O284, O298, V2335
\CYRUSHRT . . . . .	r1521	\ddag . . . . .	r313
\cyrushrt . . . . .	r1521	\ddagger . . . . .	r313, s166, s172, s179, s181, A374
\CYRV . . . . .	r1521	\ddot . . . . .	A518
\cyrv . . . . .	r1521	\ddots . . . . .	A513
\CYRW . . . . .	r1521	\deadcycles . . . . .	
\cyrw . . . . .	r1521		q334, q375, G32, G99, G165, V299
\CYRY . . . . .	r1521	debug commands:	
\cyry . . . . .	r1521	\debug_resume: . . . . .	201
\CYRYA . . . . .	r1522	\debug_suspend: . . . . .	201
\cyrya . . . . .	r1522	\DebugHooksOff . . . . .	186, <u>h1444</u> , h1550
\CYRYAT . . . . .	r1522	\DebugHooksOn . . . . .	186, <u>h1444</u> , h1549
\cyryat . . . . .	r1522	\DebugShipoutsOff . . . . .	863, <u>U403</u> , U435
\CYRYHCRS . . . . .	r1522	\DebugShipoutsOn . . . . .	863, <u>U403</u> , U434
\cyryhcrs . . . . .	r1522	\DeclareCommandCopy . . . . .	89, f460, f461, f463
		\DeclareCurrentRelease . . . . .	D812, D815, <u>S1629</u>

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

\DeclareDefaultHookRule 184, [h1448](#), h1553  
 \DeclareDocumentCommand . . . . . [g2505](#)  
 \DeclareDocumentEnvironment . . . . . [g2541](#)  
 \DeclareEmphSequence . . . . . 527  
 \DeclareEmphSequence . . . . .  
 . . . . . 528, [z541](#), z577, z578, z590  
 \DeclareEncodingSubset . . . . .  
 . . . . . [D40](#), D397, D398, D399,  
 D400, D401, D402, D403, D404,  
 D405, D406, D407, D408, D409,  
 D410, D411, D412, D413, D414,  
 D415, D416, D417, D418, D419,  
 D420, D421, D422, D423, D424,  
 D425, D427, D428, D429, D430,  
 D431, D432, D433, D434, D435,  
 D436, D437, D438, D439, D440,  
 D441, D442, D443, D444, D445,  
 D446, D447, D448, D449, D450,  
 D451, D452, D453, D454, D455,  
 D456, D457, D458, D459, D460,  
 D461, D462, D463, D464, D465,  
 D466, D467, D468, D469, D470,  
 D471, D472, D473, D474, D475,  
 D476, D477, D478, D479, D480,  
 D481, D482, D483, D484, D485,  
 D486, D487, D488, D489, D490,  
 D491, D492, D493, D494, D495,  
 D496, D497, D498, D499, D500,  
 D501, D502, D503, D504, D505,  
 D506, D507, D508, D509, D510,  
 D511, D512, D513, D514, D515,  
 D516, D517, D518, D519, D520,  
 D521, D522, D523, D524, D525,  
 D526, D527, D528, D529, D530,  
 D531, D532, D533, D534, D535,  
 D536, D537, D538, D539, D540,  
 D541, D542, D543, D544, D545,  
 D546, D547, D548, D549, D550,  
 D551, D552, D553, D554, D555,  
 D556, D557, D558, D559, D560,  
 D561, D562, D563, D564, D565,  
 D566, D567, D568, D569, D570,  
 D571, D572, D573, D574, D575,  
 D620, D820, D821, D822, D823,  
 D865, D872, D873, D874, D875,  
 D1151, D1152, D1153, D1154,  
 D1155, D1156, D1157, D1158,  
 D1159, D1160, D1161, D1162,  
 D1163, D1164, D1165, D1166,  
 D1167, D1168, D1169, D1170,  
 D1171, D1172, D1173, D1174,  
 D1175, D1176, D1177, D1178,  
 D1179, D1180, D1181, D1182,  
 D1183, D1184, D1185, D1186,  
 D1187, D1188, D1189, D1190,  
 D1191, D1192, D1193, D1194,  
 D1195, D1196, D1197, D1198,  
 D1199, D1200, D1201, D1202,  
 D1203, D1204, D1205, D1206,  
 D1207, D1208, D1209, D1210, D1211  
 \DeclareErrorFont . [u477](#), y342, z735, A49  
 \DeclareExpandableDocumentCommand [g2561](#)  
 \DeclareFixedFont . . . . . [u75](#)  
 \DeclareFontEncoding . . . . .  
 r377, r463, r716, r738, r744, r830,  
 r1039, [u118](#), A126, A127, A128, A129  
 \DeclareFontEncodingDefaults . . . . .  
 . . . . . [u168](#), x90, x91, A36  
 \DeclareFontFamily . . . . . 417, [u93](#), x85, x86  
 \DeclareFontFamilySubstitution .. [u437](#)  
 \DeclareFontSeriesChangeRule 428, v3,  
 v4, v6, v7, v8, v9, v10, v11, v12,  
 v13, v14, v15, v16, v17, v18, v19,  
 v20, v21, v22, v23, v24, v25, v26,  
 v27, v28, v29, v30, v31, v32, v33,  
 v34, v35, v36, v37, v38, v39, v40,  
 v41, v42, v43, v44, v45, v46, v47,  
 v48, v49, v50, v51, v52, v53, v54,  
 v55, v56, v57, v58, v59, v60, v61,  
 v62, v63, v64, v65, v66, v67, v68,  
 v69, v70, v71, v72, v73, v74, v75,  
 v76, v77, v78, v79, v80, v81, v82,  
 v83, v84, v85, v86, v87, v88, v89,  
 v90, v91, v92, v93, v94, v95, v96,  
 v97, v98, v99, v100, v101, v102,  
 v103, v104, v105, v106, v107, v108,  
 v109, v110, v111, v112, v113, v114,  
 v115, v116, v117, v118, v119, v120,  
 v121, v122, v123, v124, v125, v126,  
 v127, v128, v129, v130, v131, v132,  
 v133, v134, v135, v136, v137, v138,  
 v139, v140, v141, v142, v143, v144,  
 v145, v146, v147, v148, v149, v150,  
 v151, v152, v153, v154, v155, v156,  
 v157, v158, v159, v160, v161, v162,  
 v163, v164, v165, v166, v167, v168,  
 v169, v170, v171, v172, v173, v174,  
 v175, v176, v177, v178, v179, v180,  
 v181, v182, v183, v184, v185, v186,  
 v187, v188, v189, v190, v191, v192,  
 v193, v194, v195, v196, v197, v198,  
 v199, v200, v201, v202, v203, v204,  
 v205, v206, v207, v208, v209, v210,  
 v211, v212, v213, v214, v215, v216,  
 v217, v218, v219, v220, v221, v222,  
 v223, v224, v225, v226, v227, v228,  
 v229, v230, v231, v232, v233, v234,  
 v235, v236, v237, v238, v239, v240,

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lpictur.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

v241, v242, v243, v244, v245, v246,  
 v247, v248, v249, v250, v251, v252,  
 v253, v254, v255, v256, v257, v258,  
 v259, v260, v261, v262, v263, v264,  
 v265, v266, v267, v268, v269, v270,  
 v271, v272, v273, v274, v275, v276,  
 v277, v278, v279, v280, v281, v282,  
 v283, v284, v285, v286, v287, v288,  
 v289, v290, v291, v292, v293, v294,  
 v295, v296, v297, v298, v299, v300,  
 v301, v302, v303, v304, v305, v306,  
 v307, v308, v309, v310, v311, v312,  
 v313, v314, v315, v316, v317, v318,  
 v319, v320, v321, v322, v323, v324,  
 v325, v326, v327, v328, v329, v330,  
 v331, v332, v333, v334, v335, v336,  
 v337, v338, v339, v340, v341, v342,  
 v343, v344, v345, v346, v347, v348,  
 v349, v350, v351, v352, v353, v354,  
 v355, v356, v357, v358, v359, v360,  
 v361, v362, v363, v364, v365, v366,  
 v367, v368, v369, v370, v371, v372,  
 v373, v374, v375, v376, v377, v378,  
 v379, v380, v381, v382, v386, v388  
**\DeclareFontSeriesDefault** . . . . . 510  
**\DeclareFontSeriesDefault** . . . . .  
     510, 513, 516, 517, 521, 525, z34,  
     z35, z69, z71, z72, z82, z93, z102, z104  
**\DeclareFontShape** . . . . . u18,  
     u446, u447, u448, u449, u450, u451,  
     u452, u453, u454, u455, u456, u457,  
     u458, u459, u460, u461, u462, u463,  
     u464, u465, u466, x25, x27, x81, x82  
**\DeclareFontShapeChangeRule** . . v523,  
     v524, v541, v542, v543, v544, v545,  
     v546, v547, v548, v549, v550, v551,  
     v552, v553, v554, v555, v556, v557,  
     v558, v559, v560, v561, v562, v563,  
     v564, v565, v566, v567, v568, v569,  
     v570, v571, v572, v573, v574, v575,  
     v576, v577, v578, v579, v580, v581,  
     v582, v583, v584, v585, v586, v587,  
     v588, v589, v590, v591, v592, v593,  
     v594, v595, v596, v597, v601, v603  
**\DeclareFontSubstitution** . . . . .  
     . . . r745, r831, u141, A26, A34,  
     A37, A38, A130, A131, A132, A133  
**\DeclareHookRule** . . . . . 176, 179,  
     183, 184, 186, 188, h1446, h1552, G40  
**\DeclareHookrule** . . . . . 183  
**\DeclareMathAccent** . y696, A516, A517,  
     A518, A519, A520, A521, A522,  
     A523, A524, A525, A526, A527, A528  
**\DeclareMathAlphabet** . . . . .  
     . . . . . x119, x123, x125, x132,  
     y522, y685, A151, A152, A153, A154  
**\DeclareMathAlphabetCharacter** . . . y871  
**\DeclareMathDelimiter** . . . . .  
     . . . 554, y873, A255, A256, A257,  
     A258, A259, A260, A263, A265,  
     A266, A563, A565, A567, A569,  
     A571, A574, A576, A578, A580,  
     A582, A584, A586, A588, A590,  
     A592, A594, A596, A598, A600,  
     A602, A604, A606, A608, A610, A612  
**\DeclareMathRadical** . . . . . y1008, A529  
**\DeclareMathSizes** . . . . .  
     . . . . . u205, u211, u233, A157,  
     A158, A159, A160, A161, A162,  
     A163, A164, A165, A166, A167, A168  
**\DeclareMathSizes\*** . . . . . u205  
**\DeclareMathSymbol** . . 550, y809, y872,  
     y889, A169, A170, A171, A172,  
     A173, A174, A175, A176, A177,  
     A178, A179, A180, A181, A182,  
     A183, A184, A185, A186, A187,  
     A188, A189, A190, A191, A192,  
     A193, A194, A195, A196, A197,  
     A198, A199, A200, A201, A202,  
     A203, A204, A205, A206, A207,  
     A208, A209, A210, A211, A212,  
     A213, A214, A215, A216, A217,  
     A218, A219, A220, A221, A222,  
     A223, A224, A225, A226, A227,  
     A228, A229, A230, A231, A232,  
     A233, A234, A235, A236, A237,  
     A238, A239, A240, A241, A242,  
     A243, A244, A245, A246, A247,  
     A248, A249, A250, A251, A261,  
     A262, A264, A268, A269, A270,  
     A271, A272, A273, A274, A275,  
     A276, A277, A278, A279, A280,  
     A281, A282, A283, A284, A285,  
     A286, A287, A288, A289, A290,  
     A291, A292, A293, A294, A295,  
     A296, A297, A298, A299, A300,  
     A301, A302, A303, A304, A305,  
     A306, A307, A308, A309, A310,  
     A311, A312, A313, A314, A315,  
     A316, A317, A318, A319, A320,  
     A321, A322, A323, A324, A325,  
     A327, A328, A329, A330, A331,  
     A332, A333, A334, A341, A342,  
     A343, A344, A345, A346, A347,  
     A349, A350, A351, A352, A353,  
     A354, A356, A357, A358, A359,  
     A360, A361, A364, A365, A366,  
     A367, A370, A371, A372, A373,

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=lpageno.dtx, F=lxref.dtx,  
G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
X=ltfinal.dtx

A374, A375, A376, A377, A378,  
 A379, A380, A381, A382, A383,  
 A384, A385, A386, A387, A388,  
 A389, A390, A391, A392, A393,  
 A394, A395, A396, A397, A398,  
 A399, A400, A401, A402, A403,  
 A404, A405, A406, A407, A408,  
 A409, A410, A411, A414, A415,  
 A418, A419, A420, A421, A422,  
 A423, A424, A425, A426, A427,  
 A428, A429, A430, A432, A433,  
 A434, A435, A436, A437, A438,  
 A441, A442, A443, A445, A446,  
 A447, A448, A449, A450, A451,  
 A452, A453, A454, A455, A477,  
 A479, A501, A502, A503, A553,  
 A554, A555, A556, A614, A615, A616  
`\DeclareMathVersion` ..... y355, z2, z3  
`\DeclareOldFontCommand` ..... C125, C141  
`\DeclareOption` ..... 788  
`\DeclareOption` ..... r1527, w29, w37,  
 w45, w53, w56, w60, D820, D821,  
 D822, D823, D824, D826, D828,  
 D830, D831, D872, D873, D874,  
 D875, D876, D878, D879, S433, S1071  
`\DeclareOption*` ..... 788  
`\DeclareOption*` ..... S433  
`\DeclarePreloadSizes` . u185, x95, x96,  
 B19, B21, B22, B23, B25, B26, B27,  
 B28, B29, B30, B34, B38, B43, B45,  
 B49, B50, B53, B54, B57, B58, B64  
`\DeclareRelease` ..... D811, D814, S1536  
 DeclareRelease commands:  
     `\DeclareRelease:` ..... S1539  
`\DeclareRobustCommand` ..... 85, 86,  
 89, 92, 93, 95, 96, 360, 447, 617,  
f221, f728, f729, f735, f750, 14, 111,  
 130, 157, 07, 08, 09, o10, o11, o68,  
 o92, o330, o406, o420, o437, o461,  
 p2, p3, p13, q414, q527, r150, r158,  
 r307, r310, r311, r312, r313, r314,  
 r316, r318, r320, s189, t19, t20,  
 t21, u251, u279, u280, u281, u286,  
 u298, u308, u316, u318, u336, u641,  
 u650, v394, v398, v407, v409, v419,  
 v526, v531, v536, v615, v618, v626,  
 v627, v633, v713, w115, w164, z4,  
 z7, z10, z13, z16, z19, z22, z25, z28,  
 z254, z277, z307, z328, z352, z363,  
 z380, z383, z405, z410, z415, z448,  
 z453, z458, z474, z477, z480, z483,  
 z486, z500, z549, z550, z585, z591,  
 z613, z615, z624, z626, z633, z649,  
 z657, z675, z691, z708, A335, A336,  
 A337, A348, A355, A412, A413,  
 A444, A456, A460, A463, A468,  
 A470, A472, A475, A478, A480,  
 A481, A483, A485, A487, A489,  
 A491, A493, A495, A497, A499,  
 A505, A507, A509, A512, A530,  
 A533, A536, A540, A544, A547,  
 A550, A557, A560, A617, A618,  
 A619, A624, A626, A628, A630,  
 C3, C126, D4, D11, D610, D1136,  
 F47, F59, G172, G228, G357, G362,  
 G367, G377, G381, G385, G466,  
 G470, H3, H4, H5, H6, H7, H8,  
 H9, H10, H11, H12, H13, H14,  
 H15, H16, H17, H18, H19, H20,  
 H21, H22, H23, H24, H25, H26,  
 H27, H28, H29, H30, H31, H32,  
 H33, H34, H35, H39, H41, H42,  
 H43, H44, H45, H46, H47, H48,  
 H49, H50, H51, H52, H81, H82,  
 H83, H84, H126, H154, H156, H160,  
 H205, H207, H209, H211, H214,  
 H215, H217, H224, H226, H227,  
 H238, H261, H263, H279, H290,  
 H340, H341, H342, H416, H433,  
 H457, J7, J24, J120, J133, J156,  
 J157, J173, J184, J238, J428, J446,  
 J454, J490, J491, J492, J493, J494,  
 J495, K139, K142, K154, L124,  
 L127, L130, N7, N8, N9, N10,  
 N14, N222, O402, O423, Q12, R22,  
 R30, R53, R55, R57, R64, S1027,  
 S1028, S1029, S1035, S1036, S1659,  
 T148, T184, U437, V87, X551, X561  
`\DeclareSizeFunction` .....  
     ..... w417, w490, w491,  
 w502, w503, w507, w508, w514,  
 w515, w543, w557, w558, w565, w566  
`\DeclareSymbolFont` .....  
     x136, y403, A141, A142, A143, A144  
`\DeclareSymbolFontAlphabet` .....  
     ..... y1082, A148, A149, A150  
`\DeclareTextAccent` ..... r64,  
 r378, r379, r380, r381, r382, r383,  
 r384, r385, r386, r387, r388, r464,  
 r465, r466, r467, r468, r469, r470,  
 r471, r472, r473, r474, r741, r746,  
 r747, r748, r749, r750, r751, r752,  
 r753, r754, r755, r756, r838, r839,  
 r840, r841, r842, r843, r844, r845,  
 r846, r847, r848, r849, r850, r851, r852  
`\DeclareTextAccentDefault` .....  
     ..... r185, r226, r227, r228,  
 r229, r230, r231, r232, r233, r234,

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

r235, r236, r237, r238, r239, r279,  
r282, D687, D688, D940, D941,  
D942, D943, D944, D945, D946,  
D947, D948, D949, D950, D951, D952  
\DeclareTextCommand . . . . r3, r58, r65,  
r83, r389, r392, r395, r409, r410,  
r411, r414, r415, r422, r424, r426,  
r428, r434, r436, r438, r445, r475,  
r478, r482, r485, r487, r490, r492,  
r494, r510, r568, r569, r570, r730,  
r757, r759, r762, r765, r798, r805,  
r832, r835, r865, r893, r1044, r1070,  
r1072, r1074, r1156, r1158, r1160,  
r1207, r1244, D376, D377, D378,  
D379, D380, D381, D382, D383,  
D384, D385, D625, D632, D639, D759  
\DeclareTextCommandDefault r57, r186,  
r188, r283, r286, r287, r288, r290,  
r292, r296, r300, r301, r303, r304,  
r305, r306, r326, r355, D155, D157,  
D160, D162, D164, D166, D168,  
D170, D172, D174, D176, D178,  
D180, D182, D184, D186, D188,  
D190, D193, D194, D195, D196,  
D197, D198, D199, D200, D201,  
D202, D203, D204, D205, D206,  
D207, D208, D210, D212, D214,  
D216, D218, D220, D222, D224,  
D226, D228, D230, D232, D234,  
D236, D238, D240, D242, D244,  
D246, D248, D250, D252, D254,  
D256, D258, D260, D262, D264,  
D266, D268, D270, D272, D274,  
D276, D278, D280, D282, D284,  
D286, D288, D290, D292, D294,  
D296, D298, D300, D302, D304,  
D306, D309, D311, D313, D315,  
D317, D319, D321, D323, D325,  
D327, D329, D331, D333, D335,  
D337, D339, D341, D343, D345,  
D347, D349, D351, D353, D355,  
D357, D359, D361, D363, D365,  
D654, D662, D664, D670, D678,  
D1006, D1008, D1009, D1011,  
D1013, D1015, D1017, D1019,  
D1021, D1023, D1025, D1027,  
D1029, D1031, D1033, D1035,  
D1037, D1039, D1041, D1043,  
D1045, D1047, D1049, D1051,  
D1053, D1055, D1057, D1059,  
D1061, D1063, D1065, D1067,  
D1069, D1071, D1073, D1075,  
D1077, D1079, D1081, D1083,  
D1085, D1087, D1089, D1091,  
D1093, D1095, D1097, D1099,  
D1101, D1103, D1105, D1107,  
D1109, D1111, D1113, D1115,  
D1117, D1119, D1121, D1123, D1126  
\DeclareTextComposite r76, r452, r453,  
r587, r588, r589, r590, r591, r592,  
r593, r594, r595, r596, r597, r598,  
r599, r600, r601, r602, r603, r604,  
r605, r606, r607, r608, r609, r610,  
r611, r612, r613, r614, r615, r616,  
r617, r618, r619, r620, r621, r622,  
r623, r624, r625, r626, r627, r628,  
r629, r630, r631, r632, r633, r634,  
r635, r636, r637, r638, r639, r640,  
r641, r642, r643, r644, r645, r646,  
r647, r648, r649, r650, r651, r652,  
r653, r654, r655, r656, r657, r658,  
r659, r660, r661, r662, r663, r664,  
r665, r666, r667, r668, r669, r670,  
r671, r672, r673, r674, r675, r676,  
r677, r678, r679, r680, r681, r682,  
r683, r684, r685, r686, r687, r688,  
r689, r690, r691, r692, r693, r694,  
r695, r696, r697, r812, r813, r814,  
r815, r816, r817, r818, r819, r820,  
r821, r822, r823, r824, r825, r826, r827  
\DeclareTextCompositeCommand . . . . .  
. . . . . r76, r431,  
r454, r455, r456, r457, r459, r698,  
r699, r701, r704, r705, r706, r707,  
r708, r709, r710, r711, r712, r795, r1050  
\DeclareTextFontCommand C1, C15, C16,  
C17, C18, C19, C20, C21, C22,  
C23, C24, C29, C30, C31, C42, C140  
\DeclareTextSymbol . . . . r3, r398, r399,  
r400, r401, r402, r403, r404, r405,  
r406, r407, r408, r412, r413, r416,  
r417, r418, r419, r420, r421, r526,  
r527, r528, r529, r530, r531, r532,  
r533, r534, r535, r536, r537, r538,  
r539, r540, r541, r543, r544, r545,  
r546, r547, r548, r549, r550, r551,  
r552, r553, r554, r555, r556, r557,  
r558, r559, r560, r561, r562, r563,  
r564, r565, r566, r567, r571, r572,  
r573, r574, r575, r576, r577, r578,  
r579, r580, r581, r582, r583, r584,  
r585, r586, r717, r718, r719, r720,  
r721, r722, r723, r724, r725, r726,  
r727, r728, r729, r739, r740, r768,  
r769, r770, r771, r772, r773, r774,  
r776, r777, r778, r779, r780, r781,  
r782, r783, r784, r785, r786, r787,  
r788, r789, r790, r791, r792, r793,

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=lpageno.dtx, F=lxref.dtx,  
G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
X=ltfinal.dtx

r794, r853, r854, r855, r856, r857,  
r858, r859, r860, r861, r862, r863,  
r864, r876, r877, r878, r879, r880,  
r881, r882, r883, r884, r885, r886,  
r887, r888, r889, r890, r891, r892,  
r899, r900, r901, r902, r903, r904,  
r905, r906, r907, r908, r909, r910,  
r911, r912, r913, r914, r915, r916,  
r917, r918, r919, r920, r921, r922,  
r923, r924, r925, r926, r927, r928,  
r929, r930, r931, r932, r933, r934,  
r935, r936, r937, r938, r939, r940,  
r941, r942, r943, r944, r945, r946,  
r947, r948, r949, r950, r951, r952,  
r953, r954, r955, r956, r957, r958,  
r959, r960, r961, r962, r963, r964,  
r965, r966, r967, r968, r969, r970,  
r971, r972, r973, r974, r975, r976,  
r977, r978, r1076, r1077, r1078,  
r1079, r1080, r1081, r1082, r1083,  
r1084, r1085, r1086, r1087, r1088,  
r1089, r1090, r1091, r1092, r1093,  
r1094, r1095, r1096, r1098, r1099,  
r1100, r1101, r1102, r1103, r1104,  
r1105, r1106, r1107, r1108, r1109,  
r1110, r1111, r1112, r1114, r1115,  
r1116, r1117, r1118, r1119, r1120,  
r1121, r1122, r1123, r1124, r1125,  
r1126, r1127, r1128, r1129, r1130,  
r1131, r1132, r1133, r1134, r1135,  
r1136, r1137, r1138, r1139, r1140,  
r1141, r1142, r1143, r1144, r1145,  
r1146, r1147, r1148, r1149, r1150,  
r1151, r1152, r1153, r1154, r1155,  
r1162, r1163, r1164, r1165, r1166,  
r1167, r1168, r1169, r1170, r1171,  
r1172, r1173, r1174, r1175, r1176,  
r1177, r1178, r1179, r1180, r1181,  
r1182, r1183, r1184, r1185, r1186,  
r1187, r1188, r1189, r1190, r1191,  
r1192, r1193, r1194, r1195, r1196,  
r1197, r1198, r1199, r1200, r1201,  
r1202, r1203, r1205, r1206, r1218,  
r1219, r1220, r1221, r1222, r1223,  
r1224, r1225, r1226, r1227, r1228,  
D368, D369, D370, D371, D372,  
D373, D374, D375, D386, D387,  
D388, D389, D390, D391, D392,  
D393, D394, D395, D589, D590,  
D591, D592, D593, D594, D595, D596  
**\DeclareTextSymbolDefault** . . . . .  
. . . . . r185, r240, r241,  
r242, r243, r244, r245, r246, r247,  
r248, r249, r250, r251, r252, r253,  
**\DeclareUnicodeAccent** . . . . .  
r1043, r1229, r1230, r1231, r1232,  
r1233, r1234, r1235, r1236, r1237,  
r1238, r1239, r1240, r1241, r1242, r1243  
**\DeclareUnicodeCharacter** . . . . . X370  
**\DeclareUnicodeComposite** r1048, r1248,  
r1249, r1250, r1251, r1252, r1253,  
r1254, r1255, r1256, r1257, r1258,  
r1259, r1260, r1261, r1262, r1263,  
r1264, r1265, r1266, r1267, r1268,  
r1269, r1270, r1271, r1272, r1273,  
r1274, r1275, r1276, r1277, r1278,  
r1279, r1280, r1281, r1282, r1283,  
r1284, r1285, r1286, r1287, r1288,  
r1289, r1290, r1291, r1292, r1293,  
r1294, r1295, r1296, r1297, r1298,  
r1299, r1300, r1301, r1302, r1303,  
r1304, r1305, r1306, r1307, r1308,  
r1309, r1310, r1311, r1312, r1313,  
r1314, r1315, r1316, r1317, r1318,  
r1319, r1320, r1321, r1322, r1323,  
r1324, r1325, r1326, r1327, r1328,  
r1329, r1330, r1331, r1332, r1333,  
r1334, r1335, r1336, r1337, r1338,  
r1339, r1340, r1341, r1342, r1343,  
r1344, r1345, r1346, r1347, r1348,

**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=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lpictur.dtx,  
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
X=ltfinal.dtx

r1349, r1350, r1351, r1352, r1353,  
 r1354, r1355, r1356, r1357, r1358,  
 r1359, r1360, r1361, r1362, r1363,  
 r1364, r1365, r1366, r1367, r1368,  
 r1369, r1370, r1371, r1372, r1373,  
 r1374, r1375, r1376, r1377, r1378,  
 r1379, r1380, r1381, r1382, r1383,  
 r1384, r1385, r1386, r1387, r1388,  
 r1389, r1390, r1391, r1392, r1393,  
 r1394, r1395, r1396, r1397, r1398,  
 r1399, r1400, r1401, r1402, r1403,  
 r1404, r1405, r1406, r1407, r1408,  
 r1409, r1410, r1411, r1412, r1413,  
 r1414, r1415, r1416, r1417, r1418,  
 r1419, r1420, r1421, r1422, r1423,  
 r1424, r1425, r1426, r1427, r1428,  
 r1429, r1430, r1431, r1432, r1433,  
 r1434, r1435, r1436, r1437, r1438,  
 r1439, r1440, r1441, r1442, r1443,  
 r1444, r1445, r1446, r1447, r1448,  
 r1449, r1450, r1451, r1452, r1453,  
 r1454, r1455, r1456, r1457, r1458,  
 r1459, r1460, r1461, r1462, r1463,  
 r1464, r1465, r1466, r1467, r1468,  
 r1469, r1470, r1471, r1472, r1473,  
 r1474, r1475, r1476, r1477, r1478,  
 r1479, r1480, r1481, r1482, r1483,  
 r1484, r1485, r1486, r1487, r1488,  
 r1489, r1490, r1491, r1492, r1493, r1494  
 $\backslash$ def ..... 246, 402, 511, 518, 542, 653  
 $\backslash$ defaulthyphenchar ..... b367, f738, f753  
 $\backslash$ defaultscriptratio ..... u624, u631  
 $\backslash$ defaultscriptscriptratio ... u625, u631  
 $\backslash$ defaultskewchar ..... b368  
 $\backslash$ deg ..... H34  
 $\backslash$ delcode ..... y1006  
 $\backslash$ delimiter ..... y919, y989, y1000  
 $\backslash$ delimiterfactor ..... b369  
 $\backslash$ delimitershortfall ..... b379  
 $\backslash$ Delta ..... A298  
 $\backslash$ delta ..... A271  
 $\backslash$ depth ..... J32, J35  
 $\backslash$ det ..... H30  
 $\backslash$ detokenize 99, 332, 845, c93, c94, f527,  
 f545, f609, l249, l250, q253, r998,  
 r1041, v434, z563, D44, G130, S815,  
 S1143, S1147, S1275, S1276, S1280  
 $\backslash$ DH ..... r528, r1120, X571  
 $\backslash$ dh ..... r538, r1126, X571  
 $\backslash$ Diamond ..... z727  
 $\backslash$ diamond ..... A380  
 $\backslash$ diamondsuit ..... A332  
 $\backslash$ dim ..... H28

**dim commands:**  
 $\backslash$ dim\_new:N ..... U197, U198, U199, U200  
 $\backslash$ dim\_set:Nn ..... U191, U192, U193, U194  
 $\backslash$ dim\_use:N ..... U500, U501, U502  
 $\backslash$ c\_max\_dim ..... U211, U237, U262, U289  
 $\backslash$ c\_zero\_dim ..... n60,  
 U217, U218, U219, U225, U243,  
 U244, U245, U268, U269, U270,  
 U297, U298, U299, U327, U329, U330  
 $\backslash$ dimedef ..... b42, b43, b44, b52, b91, d218  
 $\backslash$ dimenzero ..... d218  
 $\backslash$ dimexpr ..... r503,  
 r519, r869, r872, r1211, r1214, L13  
 $\backslash$ directlua ..... a9, a12, a17, a20, a25, b65,  
 b81, b105, b256, b349, b359, d2,  
 d14, d29, d205, d223, d248, d253,  
 d257, f34, f649, f734, r1011, S1128,  
 S1259, X84, X86, X94, X99, X101

**disable commands:**  
 $\backslash$ disable\_callback ..... d810  
 $\backslash$ disable\_callback ..... 45  
 $\backslash$ DisableGenericHook 177, 204, h1402, h1536  
 $\backslash$ DisableHook ..... 177, 252, h1487  
 $\backslash$ DiscardShipoutBox ..... 180,  
 859–861, 864, U82, U400, U433, U489  
 $\backslash$ discretionary ..... f736, f751, f761, H238  
 $\backslash$ displaylines ..... H190  
**displaymath (environment)** ..... H332  
 $\backslash$ displaymath ..... H334  
 $\backslash$ displaystyle ..... A532, A535, A539,  
 A541, H62, H197, H359, H362,  
 H415, H440, H452, H480, H505, H508  
 $\backslash$ displaywidowpenalty ..... b327  
 $\backslash$ displaywidth ... H197, H358, H427, H483  
 $\backslash$ div ..... A383  
 $\backslash$ DJ ..... r529, r1130, X571  
 $\backslash$ dj ..... r539, r1131, X571  
 $\backslash$ do a74, a75, a126, b13, b14, f69, g1710,  
 k3, k7, k16, k26, q66, q69, q134,  
 q137, q189, q192, q232, q304, q359,  
 q498, q515, q658, q664, C90, G431,  
 G452, G560, G570, G576, G582,  
 J57, J76, K244, K269, L104, L116,  
 L123, L203, L337, L339, L362,  
 L365, L394, L396, L417, L422,  
 L478, L506, L535, L731, L787, O65,  
 O134, Q16, Q45, Q62, S230, S247,  
 S478, S497, S514, S532, S537, S551,  
 S556, S592, S602, S1060, S1097,  
 S1175, S1228, S1308, S1397, S1674  
 $\backslash$ DocInput ..... w8, A5, B5, W4  
 $\backslash$ document ..... 324  
**document (environment)** ..... G8

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

\document ..... 68, 196,  
     448, 513, 517, 616, q9, G183, Q44, Q61  
 \documentclass ..... 181, 182,  
     786, d16, w2, A2, B2, S609, S616,  
     S675, S678, S835, S969, S1080, W2  
 \documentstyle ..... S614, S1080  
 \dospecials .....  
     a74, a126, b13, g1711, G431, G452,  
     G560, G570, S1175, S1308, S1397  
 \dot ..... A525  
 \doteq ..... A469  
 \dotfill ..... b480, f776, f797  
 \dots ..... r320, r322  
 \doublehyphendemerits ..... b330  
 \doublerulesep ..... K311, K338, K362  
 \Downarrow ..... A586  
 \downarrow ..... A580  
 \downbracefill ..... A538, A557  
 \dump ..... X637

**E**

\E ..... S1182, S1185, S1213, S1315,  
     S1318, S1345, S1404, S1407, S1435  
 \edef ..... 440, 441, 489, 521  
 \egroup ..... b417  
 \eject ..... b447  
 \ell ..... A311  
 else commands:  
     \else: ..... g1155,  
         h448, h691, h698, h706, h848,  
         h1148, h1219, h1264, h1294, n29, T241  
 \em ..... 528, 529, z550, z577, C42  
 \emergencystretch ..... R59, R65  
 \emforce ..... 527  
 \emforce ..... 528, 529, z546, z573, z582  
 \eminnershape .. 527, 528, z554, z560, z577  
 \emph ..... 528, h1423, C42  
 \empty ..... b415  
 \emptyset ..... A318  
 \emreset ..... 528  
 \emreset ..... 528, z546, z549, z580  
 \emrest ..... z549  
 \encodingdefault ... 517, d252, d267,  
     d275, r990, r1528, r1561, r1562,  
     y347, z658, z676, z692, z709, A50  
 \end ..... 136, 141, 176, 195, 197,  
     198, 219, 246, 283, 616–618, 654,  
     a69, f23, f671, g1203, g1319, g1334,  
     l250, w9, A6, B6, G190, G197,  
     G199, G227, G253, G410, G411,  
     H461, H470, N15, N17, S1190,  
     S1194, S1201, S1323, S1327, S1333,  
     S1412, S1416, S1422, U385, W5, I112  
 \endarray ..... K171

\endcenter ..... G352  
 \endcsname ..... 181, 848, 850  
 \enddisplaymath ..... H335  
 \enddocument ..... 193, 197, 610,  
     612, 856, 878, 879, G8, G66, G68, U350  
 \endenumerate ..... I240  
 \endeqnarray ..... H367, H414  
 \endequation ..... H338  
 \endfilecontents ..... S1084  
 \endflushleft ..... G402  
 \endflushright ..... G404  
 \endgraf ..... 292, 299, 302, b412, n98, n136, J94  
 \endgroup ..... 326, 616, 850  
 \EndIncludeInRelease .....  
     ..... a22, a50, a306, a316,  
     b87, b101, b118, b123, b133, b137,  
     b147, b150, b167, b181, b185, b219,  
     b224, b232, b240, b288, b300, b355,  
     b363, b513, b547, b555, b580, b620,  
     b625, b635, b640, c68, d226, d249,  
     d272, d276, e10, e18, e27, e110,  
     e131, e143, e152, 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, 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, l157, l161, l204, l210,  
     o20, o30, o64, o77, o85, o90, o103,  
     o109, o151, o165, o177, o188, o205,  
     o217, o251, o268, o306, o328, o365,  
     o398, o431, o435, o444, o450, o464,  
     o471, q82, q139, q194, q255, q276,  
     q288, q346, q380, q418, q441, q464,  
     q482, q489, q508, q525, q538, q542,  
     q560, q571, q581, q620, q647, r103,  
     r123, r168, r175, r330, r352, r367,  
     r375, s28, s33, s68, s73, s98, s125,  
     s134, s176, s182, s202, s205, t10,  
     t14, u52, u73, u231, u248, u294,  
     u304, u314, u430, u435, u470, u475,  
     u492, u510, u549, u582, u698, u710,  
     v384, v390, v403, v415, v422, v505,  
     v520, v599, v611, v622, v629, v636,  
     v709, v727, v734, v738, v741, w157,  
     w160, w176, w549, w555, x21, x143,  
     y77, y105, y216, y246, y276, y308,  
     y364, y369, y745, y787, y800, y807,  
     y995, y1003, z67, z100, z122, z227,  
     z248, z300, z347, z376, z387, z434,

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

z468, z494, z530, z539, z576, z588, c75  
 z595, z629, z635, z670, z686, z703, I251  
 z718, A80, A90, A109, A118, A633, b412, H175  
 A640, C33, C40, F50, F61, F72, 613,  
 G64, G103, G111, G117, G150, a92, a93, a94, a204, f37, f39, f44,  
 G163, G225, G275, G294, G308, q595, q629, S370, S371, S372, X315  
 G317, G327, G334, G344, G349, I98, I240, I251  
 G373, G389, G398, G436, G457, J155  
 G488, G498, G516, G531, G543, H333  
 G565, G573, H86, H95, H117, J354  
 H124, H143, H147, H162, H170, \EndModuleRelease . . . . . c108, c109,  
 H219, H236, H266, H274, H303, c149, g2649, h1558, i424, n141, D809  
 H330, H394, H403, H443, H455, \Endpicture . . . . . L49  
 H464, H473, J13, J22, J69, J86, \Endsloppypar . . . . . R63  
 J101, J113, J124, J131, J188, J195, \Endtabbing . . . . . K84  
 J243, J251, J302, J320, J388, J404, \Endtabular . . . . . K171  
 J421, J430, J435, J458, J465, K64, \Endtabular\* . . . . . K171  
 K69, K156, K164, K226, K231, \Endtrivlist . . . . . G352, G402, G404,  
 L15, L19, L38, L47, L68, L76, G460, H485, K85, M39, I100, I101  
 L88, L96, L111, L121, L150, L155, \Endverbatim . . . . . G459, G486  
 L171, L182, L249, L263, L368, \Enlargethispage . . . . . V1855  
 L425, L463, L469, L500, L530, \Enlargethispage\* . . . . . V1855  
 L555, L571, L582, L595, L603, \Enskip . . . . . o473  
 L624, L641, L651, L655, L745, \Enspace . . . . . o461, o469  
 L800, L819, L836, N19, N29, N167, \Ensuremath . . . . . s178, H416, O409, O416, O437, O444  
 N173, N178, N192, N198, N224, environments:  
 N246, O104, O172, O231, O246, array . . . . . K168  
 O293, O306, O351, O368, O411, center . . . . . G351  
 O417, O426, O430, O439, O445, displaymath . . . . . H332  
 O449, O481, O498, O516, O558, document . . . . . G8  
 O564, Q56, Q72, R39, R46, S21, enumerate . . . . . I231  
 S26, S49, S61, S84, S93, S99, S111, eqnarray . . . . . H344, H486  
 S142, S149, S173, S179, S197, S208, eqnarray\* . . . . . H412  
 S242, S259, S270, S279, S296, S308, equation . . . . . H336, H474  
 S329, S342, S355, S365, S401, S417, filecontents . . . . . S1084, 786  
 S426, S458, S468, S508, S524, S546, flushleft . . . . . G401  
 S561, S575, S582, S597, S606, S640, flushright . . . . . G403  
 S647, S664, S671, S750, S777, S805, itemize . . . . . I242  
 S944, S1001, S1031, S1038, S1216, list . . . . . I34  
 S1347, S1437, T14, T23, T88, T140, lrbox . . . . . 660  
 T179, T191, T200, T245, T256, math . . . . . H332  
 T263, T291, T311, T324, T328, minipage . . . . . 661  
 T351, T371, T396, T441, T457, picture . . . . . L21  
 T464, T494, T499, U418, U455, sloppypar . . . . . R62  
 U472, U476, V53, V62, V178, V196, tabbing . . . . . K71  
 V365, V370, V418, V464, V650, tabular . . . . . K174  
 V708, V811, V830, V893, V911, thebibliography . . . . . 776  
 V953, V974, V1216, V1385, V1467, trivlist . . . . . I89  
 V1561, V1683, V1810, V1920, verbatim . . . . . G459  
 V1928, V1962, V1991, V2211, verbatim\* . . . . . G483  
 V2229, V2276, V2320, X15, X19, \epsilon . . . . . A272  
 X37, X55, X65, X72, X80, X127,  
 X132, X184, X208, X278, X283,  
 X324, X330, X426, X473, I132, I137

**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=ltspacedtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lt hyphen.dtx,  
 X=ltfinal.dtx

eqnarray (environment) . . . . . H344, H486  
 \eqnarray . . . . . H349, H413  
 eqnarray\* (environment) . . . . . H412  
 \eqno . . . . . H338  
 equation (environment) . . . . . H336, H474  
 \equation . . . . . H337  
 \equiv . . . . . A448  
 \errhelp a217, c31, l39, l66, W12, X293, X626  
 \errmessage . . . . . a4,  
     a58, a222, b164, b178, b304, c32,  
     d63, e79, e94, i47, l72, u521, u556,  
     w425, w525, x65, W16, X49, X295  
 \ERROR . . . . . 153, 154, 294,  
     g1397, g1407, g1416, g1769, g1782,  
     g1802, g1821, g1897, g1924, h834, h835  
 \errorcontextlines . b372, b503, b522,  
     b538, b553, b567, b591, b608, l212  
 \errorstopmode . . . . . 185, 188, b488, X636  
 \escapchar . . . . . 336  
 \escapechar . . . . . 94, d204,  
     f126, f169, f173, f181, f275, f276,  
     f300, f437, f525, f543, q399, u367,  
     u591, w229, y58, y86, y158, y226,  
     y256, y287, y331, T273, T296, T316  
 \eta . . . . . A274  
 \etacatcode . . . . . d862  
 \eTeXversion . . . . . a57  
 \evensidemargin . . . . . V73, V613, V672  
 \everycr . . . . . b468, H192, H195, H358, H502  
 \everydisplay . . . . . u345, u346, u351  
 \everyeof . . . . . X315  
 \everyjob . . . . . c37, c42, c47,  
     d208, d254, d255, y351, U29, U30,  
     X98, X317, X402, X596, X597, X599  
 \everymath . . . . . u344, u346, u349  
 \everypar . . . . . 286  
 \everypar . . . . . 288–293, 296–298, 302, 303,  
     654, n32, n35, n78, n88, n101, n139,  
     q43, q112, q170, u644, u657, u704,  
     G166, G433, G455, J292, J313,  
     K81, N48, N96, N107, N127, N136,  
     O187, V165, V192, V1151, V1317,  
     I129, I131, I135, I136, I180, I197  
 \EveryShipout . . . . . 864  
 \ExecuteOptions w57, w70, D843, D880, S585  
 \exhyphenpenalty . . . . . b322, b441  
 \exists . . . . . A324  
 \exp . . . . . H31  
 exp commands:  
     \exp:w . . . . . 216, h438  
     \exp\_after:wN . . . . . 216, 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, y186, y190, T73, T236, T237  
 \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, T46, T97  
 \exp\_args:Nee . . . . . T129  
 \exp\_args:Nf . . . . .  
     . . . . . g349, g1901, g1928, g1989,  
     g2164, g2248, i158, T32, T34, T361  
 \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, T53  
 \exp\_args:Nv . . . . . h983, h990, h1064  
 \exp\_args:Nx . . . . .  
     g2103, g2106, g2129, h270, h312, U29  
 \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 . . . . . 261–263, g86, g114,  
     g115, g147, g149, g150, g156, g158,  
     g199, g211, g228, g229, g230, g232,  
     g300, g815, g825, g890, 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,

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

i196, i203, i204, i208, i212, i228,  
 i269, i275, i293, i302, i354, i355,  
 i380, i381, n37, n38, U31, U95, U124  
 $\backslash\exp_not:n$  ..... 112, 260,  
 261, 263, 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, T211, T225, U30  
 $\backslash\exp_stop_f:$  g1241, g1779, h688, h696  
 expandable commands:  
      $\backslashexpandable_grab_{<type>:w}$  ..... 134  
 $\backslashexpandafter$  ..... 485, 617, 618, 848  
 expandafter commands:  
      $\backslashexpandafter$ : ..... C88, K269  
 $\backslashexpanded$  ..... 209, S815, X97  
 $\backslashExplSyntaxOff$  .....  
     ... 792, e142, g2650, h1559, i298,  
     i425, n142, y214, T12, T86, T138,  
     T243, T254, T266, T369, T394,  
     T439, T492, T547, U416, U463, U503  
 $\backslashExplSyntaxOn$  .....  
     e134, g8, h3, i3, i298, n3, y178, T7,  
     T29, T92, T207, T251, T357, T377,  
     T402, T471, T540, U5, U461, U498  
 $\backslashextracolsep$  ..... K167  
 $\backslashextrafloats$  ..... b152, b189, b274

**F**

$\backslashfam$  ..... b98, d22, d26, d38, u15  
 $\backslashfamilydefault$  ..... 516, 517, r1562, y348,  
     z444, z659, z677, z693, z710, A120  
 $\backslashfbox$  ..... 660  
 $\backslashfbox$  ..... J173, J186, J193  
 $\backslashfboxrule$  ..... J171, J207,  
     J210, J216, J218, J225, J226, X137  
 $\backslashfboxsep$  ..... J171,  
     J177, J206, J211, J221, J223, X136  
 fi commands:  
      $\backslashfi$  ..... 263, g773,  
     g776, g838, g1155, g1243, h450,  
     h692, h700, h706, h850, h1067,  
     h1129, h1150, h1221, h1266, h1296,  
     i263, i266, i271, i278, n30, n61, T241  
 $\backslashfilbreak$  ..... b445  
 file commands:  
      $\backslashg_file_curr_name_str$  ..... h1437  
      $\backslashfile_full_name:n$  ..... 844, T35

$\backslash$ file\_parse\_full\_name\_apply:nN ..  
     ..... T32, T47, T90, T93, T95  
 $\backslash$ l\_file\_search\_path\_seq ..... 836, 844, 850  
 file internal commands:  
      $\backslash$ \_\_file\_parse\_full\_name\_area:nw ..  
         ..... T102, T105, T109  
      $\backslash$ \_\_file\_parse\_full\_name\_auxi:nN ..  
         ..... T97, T100  
      $\backslash$ \_\_file\_parse\_full\_name\_base:nw ..  
         ..... T108, T112, T124  
      $\backslash$ \_\_file\_parse\_full\_name\_tidy:nnnN ..  
         ..... T119, T120, T122, T127  
 file/.../after ..... 835  
 file/.../before ..... 835  
 file/after ..... 835  
 file/before ..... 835  
 filecontents (environment) ..... S1084, 786  
 $\backslash$ filecontents ..... S1084  
 filehook internal commands:  
      $\backslash$ \_\_filehook\_clear\_replacement\_-  
         flag: ..... T384, T483  
      $\backslash$ \_\_filehook\_drop\_extension:N ..  
         ..... T44, T485  
      $\backslash$ \_\_filehook\_drop\_extension\_-  
         aux:nnn ..... T54, T57  
      $\backslash$ \_\_filehook\_file\_name\_compose:nnn ..  
         ..... T218, T373, T392  
      $\backslash$ \_\_filehook\_file\_parse\_full\_-  
         name:nN ..... 841, 851, T28,  
         T30, T53, T238, T361, T379, T381  
      $\backslash$ \_\_filehook\_file\_pop: ... T59, T489  
      $\backslash$ \_\_filehook\_file\_pop\_assign:nnm ..  
         ..... T59, T491  
      $\backslash$ \_\_filehook\_file\_push: ... T59, T487  
      $\backslash$ \_\_filehook\_file\_subst\_begin:nnm ..  
         ..... 851, T379, T389  
      $\backslash$ \_\_filehook\_file\_subst\_cycle\_-  
         error>NN ..... 852, T424, T429  
      $\backslash$ \_\_filehook\_file\_subst\_loop:NN ..  
         ..... 852, T398  
      $\backslash$ \_\_filehook\_file\_subst\_tortoise\_-  
         hare:nn ..... 851, 852, T391, T398  
      $\backslash$ \_\_filehook\_full\_name:nn ..... T30  
      $\backslash$ \_\_filehook\_if\_file\_replaced:TF ..  
         ..... 851, T384, T481  
      $\backslash$ \_\_filehook\_if\_no\_extension:nTF ..  
         ..... T44, T473  
      $\backslash$ g\_filehook\_input\_file\_seq ..... 842, T59  
      $\backslash$ l\_filehook\_internal\_tl ..... T59  
      $\backslash$ \_\_filehook\_log\_file\_record:n .. T510  
      $\backslash$ g\_filehook\_nesting\_level\_int ..  
         ..... T507, T512, T515, T516, T522  
      $\backslash$ \_\_filehook\_normalize\_file\_-  
         name:w ..... T373, T479

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltruatex.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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

\\_\_filehook\_resolve\_file\_subst:w ..... T373, T477  
 \\_\_filehook\_set\_curr\_file:nNN ..... T359, T475  
 \\_\_filehook\_set\_curr\_file\_-\_assign:nnnNN ..... T362, T364  
 \\_\_filehook\_subst\_add:nn ..... 846, T206, T208, T252  
 \\_\_filehook\_subst\_empty\_name\_-\_chk:NN ..... T208  
 \\_\_filehook\_subst\_file\_normalize:Nn ..... T208  
 \\_\_filehook\_subst\_remove:n ..... T208, T253  
 \filename@parse ..... 1, 6  
 \filesize ..... e72, e118  
 \fill ..... o454  
 \finalhyphendemerits ..... b331, G360, G364, G370  
 \firstmark ..... R51, V647, V706, V2251  
 flag internal commands:  
     \flag\_{\\_\_filehook\_file\_replaced} ..... 852  
     \flag\_{\\_\_filehook\_file\_replaced} ..... T384  
 flag commands:  
     \flag\_clear:n ..... T388  
     \flag\_if\_raised:nTF ..... T386, T407  
     \flag\_new:n ..... T384  
     \flag\_raise:n ..... T408  
 \flat ..... A328  
 \floatingpenalty ..... O469, O488, O506  
 \floatpagefraction ..... O278, V2332  
 \floatsep ..... V726, V744, V751, V2138, V2188, V2337  
 \flushbottom ..... R55  
 flushleft (environment) ..... G401  
 \flushleft ..... G401  
 flushright (environment) ..... G403  
 \flushright ..... G403  
 \fmtname ..... c1, c38, c40, c43, c45, c48, c57, S684, S688  
 \fmtversion ..... c1, c19, c38, c40, c43, c45, c48, c57, c102, c115, c163, d268, l2, r1530, z225, K1, L1, S169, S176, S701, S704, V4, X580, X606  
 \fnsymbol ..... 391  
 \fnsymbol ..... s141  
 \font ..... b473, b478, f737, f740, f752, f755, r297, r298, r299, r439, r446, r799, r806, r866, r1003, r1004, r1051, r1157, r1159, r1161, r1208, u81, u87, u89, w84, z553, z586, z592, z618, B8, B9, B10, C85, D6, D612, D626, D633, G454  
 \fontdimen ..... b473, b478, r297, r298, r299, r439, r446, r799, r806, z553, z586, z592, C85, D6, D612, D626, D633, L126, L129, L679  
 \fontencoding ..... 410, d251, d252, d274, d275, r867, r1561, u251, u286, u298, u308, y347, z658, z676, z692, A16, A24, A67, A74, A83, A85, D36  
 \fontfamily ..... 410, 437, 523, u279, z6, z9, z12, z145, z156, z482, z485, z488, z738, A58, A69, A76, A87, D28, D30, D32, D34, D85, D87, D89, D91, D916  
 \fontname ..... r1004, u89  
 \fontseries ..... 410, 437, 440, 445, 453, 518, 523, 532, 534, u279, v393, v394, v405, v407, v417, v419, z15, z18, z258, z259, z260, z261, z281, z282, z283, z284, z319, z320, z321, z322, z339, z340, z341, z342, z355, z356, z357, z358, z366, z367, z368, z369, z382, z385, z476, z479, z739  
 \fontseriesforce ..... 437, v398, v409, v420  
 \fontshape ..... 410, 445, 453, 534, r449, r809, u279, v528, v533, v538, v614, v615, v624, v626, v631, v633, v684, v688, v691, v694, v697, v700, v703, v706, v713, z21, z24, z27, z30, z740, D636  
 \fontshapeforce ..... v618, v627, v634, v714  
 \fontsize ..... p6, r302, r328, r360, r868, r1210, r1246, u79, u318, z641, z741, D104, D672, D933, O409, O416, O437, O444  
 \fontsubfuzz ..... w441, w475, G42, G82  
 \FOO ..... 74  
 \foo ..... 259  
 \footins ..... O390, O465, O484, O502, V314, V315, V316, V317, V375, V422, V482, V490, V494, V517  
 \footnote ..... 772, O452  
 \footnotemark ..... N10, O518  
 \footnoterule ..... J360, O394, V493  
 \footnotesep ..... J384, J401, J418, O451, O468, O477, O487, O495, O505, O513  
 \footnotesize ..... J376, J394, J411, O466, O485, O503  
 \footnotetext ..... N12, O535  
 \footref ..... O547  
 \footskip ..... V77, V637, V696  
 \forall ..... A323  
 \frac ..... H341  
 \frame ..... J157, J233  
 \framebox ..... 660  
 \framebox ..... J180

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

\frenchspacing ..... b398, f777, f798,  
q46, q115, q173, G459, G485, G575  
\frown ..... A451  
\fussy ..... R64  
\futurelet ..... e49,  
f677, f691, o410, o418, C83, H243, K359

**G**

\g\_@@\_\meta\_{hook}\_code\_prop ..... 202  
\Gammama ..... A297  
\gamma ..... A270  
\gcd ..... H33  
\gdef ..... 793  
gdef commands:  
  \gdef\_ ..... H256  
\ge ..... A417, A419  
\GenericError ..... l18, 185, l111, l164, w62  
\GenericInfo .....  
  .... c103, c105, c116, c119, c124,  
  c159, c160, c165, c169, c173, c182,  
  e63, l4, l104, l130, l182, w31, w34,  
  w39, w75, S66, S74, S104, S107, S1600  
\GenericWarning .. l11, l94, l120, l141,  
  l149, l173, l195, w42, w47, w50, w78  
\geq ..... A415, A417  
\GetDocumentCommandArgSpec ..... g2631  
\GetDocumentEnvironmentArgSpec .. g2631  
\GetFileInfo ..... A3  
\gets ..... A439, A441  
\gg ..... A433  
\globaldefs ..... u593,  
  w231, y60, y89, y160, y228, y258, y290  
\glossary ..... 773  
\glossary .....  
  N182, P23, P35, R24, R32, V621, V680  
\glossaryentry ..... P32  
\goodbreak ..... b445, f778, f799  
\grave ..... A517  
group commands:  
  \group\_align\_safe\_begin/end: ... 148  
  \group\_align\_safe\_begin: .....  
    .... 111, 114, g218, g230, g335  
  \group\_align\_safe\_end: .....  
    .... 111, 148, g252, g358, g1621  
  \group\_begin: ..... g1614, g1971,  
  g1976, g2095, g2099, g2127, h225,  
  i198, n15, n21, y200, T210, T224, U92  
  \c\_group\_begin\_token .....  
    .... g595, g606, g1619, g1700, g2214  
  \group\_end: ..... g1634,  
  g1980, g1985, g2109, g2110, g2112,  
  g2114, g2124, g2131, g2132, h228,  
  i202, n17, n25, y203, T220, T232, U94  
  \c\_group\_end\_token ..... g598, g1689

\group\_insert\_after:N .....  
  .... 485, 486, y186, y187, y190, y211, y212  
\guillemetleft ..... r540, r773, r1096  
\guillemetright ..... r541, r774, r1112  
\guillemotleft ..... r543, r776, r1098  
\guillemotright ..... r544, r777, r1114  
\guilsinglleft ..... r545, r1175  
\guilsinglright ..... r546, r1176

**H**

\H ..... l24, r230, r385, r469,  
r601, r609, r628, r636, r753, r1238,  
r1377, r1378, r1405, r1406, D177, D201  
\halign . 84, b468, H127, H197, H358, H502  
\hangindent ..... N139  
\hat ..... A523  
\hbadness ..... b318, u647, u654,  
  u689, u708, U236, U238, U288, U290  
\hbar ..... A335  
\hbox ..... 299, 858, 860, 863  
hbox commands:  
  \hbox:n ..... U371  
  \hbox\_set:Nn .....  
    .... U168, U215, U241, U266, U295  
  \hbox\_set\_to\_wd:Nnn U239, U291, U327  
  \hbox\_unpack:N ..... U249, U293  
  \hbox\_unpack\_drop:N ..... U178  
\headheight ..... V75, V626, V685  
\headsep ..... V76, V635, V694  
\heartsuit ..... A333  
\height ..... r872, r1214, J31, J34  
\hfil ..... 646  
\hfuzz ..... b373, u655, R60, R61,  
  R67, R68, U235, U237, U287, U289  
\hglue ..... b435  
\hideoutput ..... b626  
\hideskip ..... b309, b459  
\hidewidth ..... b459, r327,  
  r329, r358, r362, r390, r391, r394,  
  r397, r476, r477, r481, r484, r486,  
  r489, r501, r506, r522, r760, r761,  
  r764, r767, r834, r837, r1245, r1247  
\hline ..... K358, K361  
\holdinginserts ..... b335  
\hom ..... H29  
hook commands:  
  \hook\_activate\_generic:n .....  
    .... 187, 207, h145, h165, h167, h1400,  
    h1401, h1405, h1467, h1486, h1499  
  \hook\_debug\_off: ..... 188, h7, h1445  
  \hook\_debug\_on: ..... 188, h7, h1444  
  \hook\_disable:n ..... h1455  
  \hook\_disable\_generic:n .....  
    .... 186, h121, h1402, h1460, h1492

**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=ltSPACE.dtx, p=ltLOGOS.dtx, q=ltFILES.dtx,  
r=ltOUTENC.dtx, s=ltCOUNTS.dtx, t=ltLENGTH.dtx, u=ltFSSBAS.dtx, v=ltFSSAXES.dtx,  
w=ltFSSTRC.dtx, x=ltFSSCMP.dtx, y=ltFSSDCL.dtx, z=ltFSSINI.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltFNTCMD.dtx, D=ltTEXTCOMP.dtx, E=ltPAGENO.dtx, F=ltXREF.dtx,  
G=ltMISCEN.dtx, H=ltMATH.dtx, I=ltLISTS.dtx, J=ltBOXES.dtx, K=ltTAB.dtx, L=ltPICTUR.dtx,  
M=ltTHM.dtx, N=ltSECT.dtx, O=ltFLOAT.dtx, P=ltIDXGLO.dtx, Q=ltBIBL.dtx, R=ltPAGE.dtx,  
S=ltCLASS.dtx, T=ltFILEHOOK.dtx, U=ltSHIPOUT.dtx, V=ltOUTPUT.dtx, W=ltHYPHEN.dtx,  
X=ltFINAL.dtx

```

\hook_gclear_next_code:n 187, h1111, h1415
\hook_gput_code:n 205
\hook_gput_code:nnn 187, 212, 214, h323, h376, h393, h1411
\hook_gput_next_code:nn .. 187, 214, h383, h398, h1086, h1413
\hook_gremove_code:nnn 188, 220, h601, h1417
\hook_gset_rule:nnnn 188, 221, h636, h1447, h1449, h1452
\hook_if_empty:nTF 180, 181, 188, h963, h1225, h1453, U63, U162
\hook_if_empty_p:n 188, h1008, h1225, U67, U113, U366
\hook_log:n 188, h934, h1443
\hook_new:n 186, 187, 204, 205, 207, 215, 240, 241, h68, h108, h506, h1396, U152, U153, U154, U155, U156, U157
\hook_new_pair:nn 186, h107, h1398, n6, n7
\hook_new_reversed:n 186, h100, h108, h1397
\g_hook_patch_action_list_tl ... 16, i90, i123
\hook_provide:n h1455
\hook_provide_pair:nn h1455
\hook_provide_reversed:n h1455
\hook_show:n ... 188, 232, h934, h1442
\hook_use:n 180, 181, 186, 187, 191, 204, 206, 210, 236, 238, 239, h724, h1115, h1440, n20, n28, n56, n63, U62, U65, U71, U75
\hook_use_once:n 180, 187, 191, 238, 239, h1191, h1441
hook internal commands:
\g_hook_??_code_prop h633
\g_hook_??_reversed_tl h633
\g_hook_<hook>_code_prop 202, 220, 240
\g_hook_<hook>_labels_clist ... 205
\g_hook_<hook>_reversed_tl ... 202
__hook_activate_generic:n ... h145
__hook_activate_generic:nn ... 148, h149, h1484
__hook_activate_generic_pair:nn h1455, h1513
__hook_activate_generic_- reversed:n h1455, 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 ... 227, 229, 235, h781, h782, h836
__hook_apply_rule:nnn 229, h846, h852
__hook_apply_rule:nnnN 229
__hook_apply_rule_->:nnn ... h887
__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 139, h1216, h1261
__hook_clear_next:n ... h1108, h1111
__hook_clist_gput:Nn h745, h747, h805, h834
__hook_cmd_begindocument_code: ... 268, i45, i422
__hook_cmd_if_scansable:NnTF ... 265, 267, i323, i345
__hook_cmd_patch_xparse:Nnn ... i127, i148
__hook_cmd_try_patch:nn ... i51, i62
\l_hook_cur_hook_tl 231, h29, h767, h893, h904
__hook_curr_name_pop: 211, h255, h1433, h1527
__hook_curr_name_push:n 211, 245, h255, h1431, h1436
__hook_curr_name_push_aux:n .. h255
__hook_currname_or_default: ... 208, h173, h181, h185, h201, h202, h350, h1326, h1356
__hook_debug:n 199, h7, 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
__hook_debug_label_data:N h786, h827, h922
__hook_debug_print_rules:n .. h1070
__hook_declare_deprecated_- generic:NNn h437, h458
__hook_declare_deprecated_- generic:NNw h453
__hook_def_cmd:w 261, i12, i189, i195, i220
\g_hook_delayed_patches_prop ... 16, i50, i56, i57
__hook_DEPRECATED_generic_- warn:n 216, h436, h443, h627, h645, h952

```

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

```

__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
__hook_DISABLE:n
 ... h121
__hook_DO_DEPRECATED_GENERIC:Nn
 h453, h628, h646, h953
__hook_DO_DEPRECATED_GENERIC:Nw
 h453
__hook_DOUBLE_HASHES:n
 ... 260, 262, i185, i246
__hook_DOUBLE_HASHES:w ...
 ... 262, i246
__hook_DOUBLE_HASHES_GROUP:n ...
 ... i246
__hook_DOUBLE_HASHES_OUTPUT:N ...
 ... 263, i246
__hook_DOUBLE_HASHES_SPACE:w ...
 ... i246
__hook_DOUBLE_HASHES_STOP:w ...
 ... i246
__hook_END_DOCUMENT_LABEL_-
 check: h255
__hook_EXP_NOT:n
 ... 260, i168, i169, i173, i184
__hook_EXP_NOT:NN
 ... i12, i190, i196, i212, i214
__hook_FILE_HOOK_NORMALIZE:n ...
 ... 218, h405, h554, h1178
\l_HOOK_FRONT_TL h758, h797, h800,
 ... h803, h805, h806, h807, h820, h821
\c_HOOK_GENERIC_{type}./{place}_TL
 216
\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
\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_ii_-
 prop h484, h507, h592
\c__hook_generics_reversed_iii_-
 prop h487, h510, h592
__hook_gput_code:nnn
 ... h323
__hook_gput_next_code:nn
 ... h1087, h1088
__hook_gput_next_do:nn
 ... 214, h384, h398, h1088
__hook_gput_next_do:Nnn
 ... h1088
__hook_gput_undeclared_hook:nnn
 214, h366, h377, h393
__hook_gremove_code:nn
 ... h601
__hook_gset_rule:nnnn
 ... h636
\c__hook_hash_tl 260,
 ... 262, 263, i11, i176, i177, i179, i264
\g__hook_hook_curr_name_tl
 209-212,
 ... 245, h32, h204, h214, h255, h267,
 ... h282, h283, h290, h300, h301, h321
__hook_hook_gput_code_do:nnn ...
 h115, h323, h369
__hook_if_declared:nTF
 ... 203, 204, h72, h154, h506, h1253, i66
__hook_if_declared_p:n
 ... h1253
__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:nTF
 ... 203, 204, 207,
 ... h121, h151, h336, h961, h1034, h1090
__hook_if_disabled_p:n
 ... h121
__hook_if_EXECUTE_IMMEDIATELY:nTF
 h327, h1193, h1212
__hook_if_EXECUTE_IMMEDIATELY_-
 p:n h1212
__hook_if_file_hook:wtf
 ... 215, 217, 219, 238, h402, h520, h1175
__hook_if_file_hook_p:w
 ... h520
__hook_if_GENERIC:nTF
 h421, h525, h958, h1268

```

**File Key:** a=ltDIRCHK.dtx, b=ltPLAIN.dtx, c=LTVERS.dtx, d=LTLUATEX.dtx, e=LTEXPL.DTX,  
f=LTDFNS.dtx, g=LTCMD.dtx, h=LTHOOKS.dtx, i=LTCDMDHOOKS.dtx, j=LTALLOC.dtx, k=LTCTRL.dtx,  
l=LTERROr.dtx, m=LTPAR.dtx, n=LTPARA.dtx, o=LTSpace.dtx, p=LTLGOS.dtx, q=LTFILES.dtx,  
r=Ltoutenc.dtx, s=LTCOUNTS.dtx, t=LTLLENGTH.dtx, u=LTFSSBAS.dtx, v=LTFSSAXES.dtx,  
w=LTFSSTRC.dtx, x=LTFSSCMP.dtx, y=LTFSSDCL.dtx, z=LTFSSINI.dtx, A=FONTDEF.dtx,  
B=PRELOAD.dtx, C=LTFNTCMD.dtx, D=LTEXTCOMP.dtx, E=LTPAGENO.dtx, F=LTXREF.dtx,  
G=LTMISCEN.dtx, H=LTMATH.dtx, I=LTLISTS.dtx, J=LTDBOXES.dtx, K=LTTAB.dtx, L=LTPICTUR.dtx,  
M=LTHM.dtx, N=LTSect.dtx, O=LTFLOAT.dtx, P=LTIIDXGLO.dtx, Q=LTBIBL.dtx, R=LTPAGE.dtx,  
S=LTCCLASS.dtx, T=LTFILEHOOK.dtx, U=LTHSHIPOUT.dtx, V=LTOUPUT.dtx, W=LTHYPHEN.dtx,  
X=LTFINAL.dtx

```

__hook_if_generic:w ... h1269, h1270
__hook_if_generic_p:n h1268
__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:nTF ...
..... 260, 262, i166, i232
__hook_if_has_hash:w i232
__hook_if_has_hash_check:w ... i232
__hook_if_has_hash_p:n i232
__hook_if_label_case:nnnn ...
..... h702, h779, h1048
__hook_if_public_command:N ... 257
__hook_if_public_command:NTF ... i76
__hook_if_public_command:w ... i76
__hook_if_reversed:nTF ...
..... h743, h977, h1014, h1016, h1259
__hook_if_reversed_p:n h1259
__hook_if_structure_exist:nTF ...
..... 203, 204,
h93, h605, h1093, h1227, h1247, h1387
__hook_if_structure_exist_p:n h1247
__hook_if_usable:nTF ...
..... 203, 204, 220,
h330, h345, h423, h478, h621, h732,
h959, h976, h1032, h1214, h1241, h1454
__hook_if_usable_p:n .. h1007, h1241
__hook_if_usable_use:n ... 238, h1162
__hook_include_legacy_code_-
chunk:n h88, h109, h731
__hook_init_structure:n ...
..... 203, 205, 206,
213, 240, h85, h91, h352, h368, h650
__hook_initialize_all: .. h710, h1525
__hook_initialize_hook_code:n ...
..... 229, h711, h727, h1132, h1159
__hook_initialize_single:NNn ...
..... 223, 225, 226, h749, h763
\l__hook_label_0_t1 h758
__hook_label_if_exist_apply:nnnTF
..... 229, 235, h836
__hook_label_ordered:nn 223
__hook_label_ordered:nnTF ...
..... 222, h663, h669, h675, h694
__hook_label_ordered_p:nn ... h694
__hook_label_pair:nn ...
..... 222, 223, h662, h668, h674,
h678, h681, h685, h686, h918, h919
\l__hook_labels_int ...
... 228, h758, h766, h770, h802, h823
\l__hook_labels_seq ...
..... h758, h765, h771, h789, h924
__hook_list_if_rule_exists:nnnTF
..... h1041
__hook_list_one_rule:nnn ... h1041
__hook_list_rules:nn ...
..... 234, h994, h1041, h1075
__hook_log:nN h934
__hook_log_cmd:n ...
..... h936, h941, h945, h947, h957
__hook_log_line:n h934
__hook_log_line_indent:n ... h934
__hook_log_next_code:n .. h990, h1039
__hook_make_name:n ... 202,
210, h195, h201, h210, h216, h270, h312
__hook_make_name:w h216
__hook_make_prefixes:w ... i153
__hook_make_usable:n ...
..... 215, h76, h79, h157, h427, h482
__hook_msg_pair_found:nnn h861,
h868, h875, h883, h891, h902, h915
\g__hook_name_stack_seq ...
..... h32, h256, h257,
h261, h268, h282, h289, h297, h307
__hook_new:n h68, h104
__hook_new_reversed:n h100
__hook_next_<(hook) ... 203, 220, 240
__hook_normalize_hook_args:Nn ...
..... h69, h101, h124, h148, h223,
h937, h942, h1087, h1112, h1194, h1484
__hook_normalize_hook_args:Nnn ...
..... h223, h324, h602
__hook_normalize_hook_args_-
aux:Nn ... h223
__hook_normalize_hook_rule_-
args:Nnnnn ... h223, h638
\l__hook_param_text_t1 ...
..... 260, 261, i8, i172, i193, i221
__hook_parse_dot_label:n h174, h176
__hook_parse_dot_label:w ... h176
__hook_parse_dot_label_aux:w . h176
__hook_parse_dot_label_cleanup:w ...
..... h176
__hook_parse_label_default:n ...
..... h170,
h235, h241, h242, h249, h250, h252
__hook_patch_check:NNnn ... i76
__hook_patch_cmd_or_delay:Nnn ...
..... 215, 255, 256, i28, i41, i45
__hook_patch_command:Nnn ...
..... 256, i55, i73, i76
__hook_patch_debug:n ...
..... i17, i78, i79, i82,
i85, i88, i155, i286, i322, i325, i326, i331

```

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
X=ltfinal.dtx

```

__hook_patch_DeclareRobustCommand:Nnn
 258, i125, i129
__hook_patch_DeclareRobustCommand_-
 aux:Nnn i131, i134
__hook_patch_expand_redefine:NNnn
 . 258, 264, i140, i145, i150, i153, i288
__hook_patch_newcommand:Nnn ...
 258, i126, i139, i143
\l__hook_patch_num_args_int ...
 i7, i156, i161, i164, i178
\l__hook_patch_prefixes_tl ...
 261, i8, i203, i219
__hook_patch_required_catcodes:
 267, i336, i357, i383
__hook_patch_retokenize:Nnnn ...
 265, i327, i362
__hook_preamble_hook:n 236,
 238, h725, h956, h1115, h1186, h1198
\l__hook_rear_tl ...
 .. h758, h787, h793, h794, h816, h817
__hook_redefine_with_hooks:Nnnn
 261, i153
\l__hook_replace_text_tl . 260, i8,
 i173, i180, i181, i182, i187, i194, i214
__hook_retokenize_patch:Nnn i91, i284
\l__hook_return_tl ...
 h25, h289, h290, h297,
 h301, h358, h361, h617, h803, h804
__hook_rule_<_gset:nnn h660
__hook_rule_>_gset:nnn h660
__hook_rule_after_gset:nnn .. h660
__hook_rule_before_gset:nnn ...
 227, h660
__hook_rule_gclear:nnn ...
 223, h651, i683
__hook_rule_incompatible-error_-
 gset:nnn h677
__hook_rule_incompatible-warning_-
 gset:nnn h677
__hook_rule_unrelated_gset:nnn .
 223, h683
__hook_rule_voids_gset:nnn .. h672
__hook_seq_csname:n ...
 .. h756, h773, h807, h864, h871, h929
__hook_set_default_hook_label:n
 h305, h1419
__hook_set_default_label:n .. h305
__hook_str_compare:nn ...
 h23, h688, h696, h705
__hook_strip_double_slash:n . h554
__hook_strip_double_slash:w . h554
__hook_tl_csname:n ...
 .. h756, h762, h772, h788, h791,
 h793, h797, h809, h811, h814, h816,
 h821, h862, h863, h869, h870, h928
__hook_tl_gclear:N ...
 h65, h117, h610,
 h611, h615, h798, h1207, h1208, h1209
__hook_tl_gput:Nn ...
 228, h744, h746, h804, h830, h834
__hook_tl_gput_left:Nn ... h58, h744
__hook_tl_gput_right:Nn ...
 h55, b353, h746, h831, h1109
__hook_tl_gset:Nn ...
 h46, h56, h60, h662, h668,
 h674, h678, h681, h736, h1108, h1204
__hook_tl_gset_eq:NN h64, h66
__hook_tl_set:Nn 201, h40, h772
__hook_tmp:w ...
 267, h34,
 h258, h262, h264, h1043, h1064,
 h1073, h1084, i170, i176, i177,
 i179, i348, i351, i355, i365, i368, i381
\l__hook_tmpt_bool ...
 .. 233, 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
__hook_toplevel<(hook) ...
 203, 213, 220, 240
__hook_try_declar ing_generic_-
 hook:nnn 214, h338, h371
__hook_try_declar ing_generic_-
 hook:nNnn 214, 215, h392, h397, h400
__hook_try_declar ing_generic_-
 hook:wnTF 216, h375, h382, h411, h416
__hook_try_declar ing_generic_-
 hook_split:nNnn h400
__hook_try_declar ing_generic_-
 next_hook:nn .. 214, h371, h1095
__hook_try_file_hook:n .. 238, h1162
__hook_try_patch_with_catcodes:Nnnn
 264, i302, i319
__hook_try_put_cmd_hook:n ...
 h426, h481, i20
__hook_try_put_cmd_hook:w i20
__hook_unpatchable_cases:n i407, i409
__hook_update_hook_code:n ...
 213, 220, 224, h333,
 h622, h655, h709, h711, h715, h1106
__hook_use:wn 237, h1143, h1156, h1162
__hook_use_end: h1115
__hook_use_initialized:n ...
 236, h724, h1115, h1200
__hook_use_once:n h1191
__hook_use_once_clear:n ...
 239, h1201, h1203

```

**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=ltSPACE.dtx, p=ltLOGOS.dtx, q=ltFILES.dtx,  
r=ltOUTENC.dtx, s=ltCOUNTS.dtx, t=ltLENGTH.dtx, u=ltFSSBAS.dtx, v=ltFSSAXES.dtx,  
w=ltFSSTRC.dtx, x=ltFSSCMP.dtx, y=ltFSSDCL.dtx, z=ltFSSINI.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltFNTCMD.dtx, D=ltTEXTCOMP.dtx, E=ltPAGENO.dtx, F=ltXREF.dtx,  
G=ltMISCEN.dtx, H=ltMATH.dtx, I=ltLISTS.dtx, J=ltBOXES.dtx, K=ltTAB.dtx, L=ltPICTUR.dtx,  
M=ltTHM.dtx, N=ltSECT.dtx, O=ltFLOAT.dtx, P=ltIDXGLO.dtx, Q=ltBIBL.dtx, R=ltPAGE.dtx,  
S=ltCLASS.dtx, T=ltFILEHOOK.dtx, U=ltSHIPOUT.dtx, V=ltOUTPUT.dtx, W=ltHYPHEN.dtx,  
X=ltFINAL.dtx

\_hook\_use\_once\_set:n . . . . .	299
. . . . . 239, h1199, h1203	
\_hook\_use\_undefined:w . . . . .	525
\g\_hook\_used\_prop . . . . .	525
. . . . . h31, h712, h719, h752	525
\l\_hook\_work\_prop . . . . .	525
. . . . . 227, h30, h748, h768, h775,	525
h777, h786, h803, h827, h896, h907	525
hook_\? internal commands:	
\_\_hook_\??. . . . .	221
hook_\<hook> internal commands:	
\_\_hook_\<hook> . . . . .	202, 203, 207, 225
\hookleftarrow . . . . .	A480
\hookrightarrow . . . . .	A478
hook ?? internal commands:	
\_\_hook~??. . . . .	h633
\phantom . . . . .	H75
\hrule . . . . .	b436,
b480, o350, o358, o384, o392, r289,	
r294, A340, A618, J163, J168, J216,	
J226, K359, K376, L590, L600, O395	
\hrulefill . . . . .	b480, f779, f800
\hspace . . . . .	o437, o441, o446
\hss . . . . .	877
\Hwithstroke . . . . .	364, r494, r1205
\hwithstroke . . . . .	364, r510, r1206
\hyphenation . . . . .	r205
\hyphenchar . . . . .	100,
f737, f740, f749, f752, f755, f760, G454	
\hyphenpenalty . . . . .	b321, u661, u693
<b>I</b>	
\I . . . . .	b404, S1209,
S1341, S1431, S1451, X241, X541	
\i . . . . .	r247, r402, r452, r453, r454, r455,
r456, r457, r547, r587, r588, r680,	
r682, r684, r686, r778, r1132, r1287,	
r1289, r1291, r1293, r1344, r1347,	
r1350, r1353, r1423, X245, X545, X552	
\ialign . . . . .	b468, b470,
A337, A459, A530, A533, A537,	
A540, H155, H157, H176, K191, L141	
\IeC . . . . .	X343, X347, X454
\if . . . . .	613
if commands:	
\if:w . . . . .	239, h1217, h1262
\if_case:w . . . . .	g1241, h688, h705
\if_catcode:w . . . . .	i263, 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, T241
\if_mode... . . . . .	299
\if_mode_horizontal: . . . . .	n29
\IfBold . . . . .	525
\IfBooleanF . . . . .	g2600, s82, s90
\IfBooleanT . . . . .	g2600
\IfBooleanTF . . . . .	173, g2600
\IfClassAtLeastTF . . . . .	788
\IfClassAtLeastTF . . . . .	S165
\IfClassLoadedTF . . . . .	788
\IfClassLoadedTF . . . . .	S261
\IfClassLoadedWithOptionsTF . . . . .	788
\IfClassLoadedWithOptionsTF . . . . .	S261
\ifcsname . . . . .	
. . . . . 439, 446, f635, f652, v440, v443,	
v653, v656, w128, y153, z42, z53,	
z75, z86, S1098, S1229, U422, V1903	
\ifdefined . . . . .	b504, b568, e22, e71, e72,
e73, e74, e117, e118, e119, z623, X305	
\iff . . . . .	A500
\IfFileExists . . . . .	324, 789
\IfFileExists . . . . .	342, a178, e37,
e68, e114, q414, q426, q528, q584,	
q609, S841, T149, T185, T195, X574	
\iffontchar . . . . .	
. . . . . 866, r1051, r1157, r1159, r1161, r1208	
\IfFontSeriesContextTF . . . . .	
. . . . . 525, z496, z532, z534	
\IfFormatAtLeastTF . . . . .	788
\IfFormatAtLeastTF . . . . .	619, 846, S165
\IfHookEmptyTF . . . . .	
. . . . . 180, 181, 184, 246, h1453, h1556, G210	
\IfHookExistsTF . . . . .	246, h1454, h1555
\ifinner . . . . .	H264,
H272, H292, H319, O57, O126, O315	
\IfNoValueF . . . . .	g2619
\IfNoValueT . . . . .	g2619
\IfNoValueTF . . . . .	g2619
\ifnum . . . . .	862
\ifodd . . . . .	y1064, L320,
L344, L378, L400, O68, O137, V21,	
V138, V610, V668, V982, V985,	
V1018, V1021, V1132, V1135,	
V1294, V1297, V1574, V1577,	
V1695, V1698, V1818, V2073, V2081	
\IfPackageAtLeastTF . . . . .	788
\IfPackageAtLeastTF . . . . .	S165
\IfPackageLoadedTF . . . . .	788
\IfPackageLoadedTF . . . . .	S261
\IfPackageLoadedtTF . . . . .	S264, S272
\IfPackageLoadedWithOptionsTF . . . . .	788
\IfPackageLoadedWithOptionsTF . . . . .	S261
\IfPDFManagementActiveTF . . . . .	977, X633
\IfTargetDateBefore . . . . .	S1659

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltruatex.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=ltspaced.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

\IfValueF ..... g2622  
 \IfValueT ..... g2622  
 \IfValueTF ..... g2622  
 \ifvbox ... V319, V376, V423, V502, V518  
 \ifx ..... 99, 262, 513, 515, 529, 848  
 \ignorespaces ..... 196, 352, o49,  
     o143, o162, o174, o185, o201, o214,  
     o483, q71, q138, q193, r72, u292,  
     u302, u312, G216, G223, G274,  
     G289, G337, G342, G348, H300,  
     H327, J154, J384, J401, J418,  
     K57, K62, K68, K83, K92, K105,  
     K109, K116, K123, K125, K134,  
     K154, K237, K301, K303, K305,  
     K332, L36, L46, L66, L75, L109,  
     L120, L131, L143, L148, L154,  
     M30, M32, N110, O17, O24, O477,  
     O495, O513, Q7, Q9, U328, I55, I217  
 \ignorespacesafterend ..... G7  
 \IJ ..... r250, r436, r550, r1133, X571  
 \ij ..... r249, r434, r549, r1134, X571  
 \Im ..... A314  
 \imath ..... A309  
 \immediate ..... 197  
 \in ..... A429, A461  
 in commands:  
     in\_callback ..... d794  
 \in\_callback ..... 44  
 \include ..... 324  
 \include . 194, 196, 334, 835, 837, 838,  
     842, q217, q265, q267, q283, q285, q340  
 include/.../after ..... 838  
 include/.../before ..... 838  
 include/.../end ..... 838  
 include/after ..... 838  
 include/before ..... 838  
 include/end ..... 838  
 \IncludeInRelease .....  
     71, 103, a18, a23, a290, a307, b49,  
     b88, b103, b119, b125, b134, b139,  
     b148, b154, b168, b182, b186, b220,  
     b228, b233, b244, b289, b347, b357,  
     b491, b515, b549, b556, b582, b622,  
     b627, b637, c68, c177, d3, d227,  
     d250, d273, e2, e11, e20, e58, e112,  
     e132, e144, 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, g2646, 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, o5, o21, o54, o65,  
     o81, o86, o98, o104, o132, o152,  
     o166, o178, o191, o206, o235, o252,  
     o271, o307, o333, o366, o426, o432,  
     o440, o445, o459, o465, q10, q83,  
     q141, q219, q256, q277, q292, q348,  
     q409, q419, q445, q465, q483, q493,  
     q509, q534, q539, q547, q561, q572,  
     q588, q621, r77, r104, r148, r169,  
     r324, r332, r353, r369, s24, s30,  
     s46, s69, s77, s99, s126, s161, s177,  
     s185, s203, t5, t11, u24, u53, u210,  
     u232, u284, u295, u305, u420, u431,  
     u439, u471, u479, u493, u514, u550,  
     u637, u699, v2, v385, v392, v404,  
     v416, v424, v506, v522, v600, v613,  
     v623, v630, v638, v710, v729, v735,  
     v739, w113, w158, w161, w541,  
     w550, x2, x22, y49, y78, y138,  
     y217, y247, y278, y357, y365, y698,  
     y746, y792, y801, y985, y996, z33,  
     z68, z101, z126, z228, z252, z301,  
     z348, z377, z395, z435, z469, z498,  
     z531, z543, z577, z590, z621, z630,  
     z655, z672, z688, z705, A63, A81,  
     A100, A110, A622, A634, C27, C34,  
     D607, F38, F51, F62, G10, G65,  
     G107, G112, G124, G151, G170,  
     G226, G276, G301, G309, G322,  
     G328, G340, G345, G355, G374,  
     G391, G415, G437, G464, G489,  
     G502, G517, G533, G556, G566,  
     H79, H87, H109, H118, H139, H144,  
     H152, H163, H203, H220, H259,  
     H267, H277, H304, H386, H395,  
     H432, H444, H456, H465, J4, J14,  
     J50, J70, J91, J102, J117, J125,  
     J181, J189, J235, J244, J281, J303,  
     J371, J389, J405, J425, J431, J451,  
     J459, K60, K65, K137, K157, K222,  
     K227, L10, L16, L26, L39, L55,  
     L69, L80, L89, L100, L112, L146,  
     L151, L160, L172, L235, L250,  
     L311, L369, L455, L464, L473,  
     L501, L531, L558, L572, L585,  
     L596, L608, L625, L645, L652,  
     L686, L746, L804, L820, N5, N20,  
     N161, N168, N174, N186, N193,  
     N202, N225, O35, O105, O206,  
     O232, O280, O294, O335, O352,  
     O406, O412, O420, O427, O434,

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

O440, O446, O463, O482, O499,  
 O549, O559, Q41, Q57, R20, R40,  
 S18, S23, S36, S51, S63, S87, S95,  
 S101, S126, S144, S167, S174, S185,  
 S198, S224, S243, S263, S271, S282,  
 S298, S312, S330, S343, S359, S379,  
 S402, S418, S447, S459, S491, S509,  
 S529, S547, S566, S576, S586, S598,  
 S630, S641, S655, S665, S721, S751,  
 S778, S810, S945, S1025, S1032,  
 S1086, S1217, S1348, T5, T15,  
 T27, T89, T144, T181, T192, T205,  
 T249, T257, T269, T292, T312,  
 T325, T332, T355, T375, T400,  
 T446, T458, T469, T496, U3, U419,  
 U459, U473, V24, V54, V151, V179,  
 V345, V366, V371, V419, V591,  
 V651, V794, V812, V873, V894,  
 V930, V954, V1066, V1217, V1386,  
 V1468, V1562, V1684, V1898,  
 V1921, V1935, V1963, V2194,  
 V2212, V2231, V2277, X8, X16,  
 X23, X38, X57, X66, X73, X92,  
 X129, X152, X185, X274, X279,  
 X300, X325, X334, X427, I125, I133  
 \includeonly ..... 324  
 \includeonly ..... 332, 838, q217, q257, q259, q278, q279  
 \indent ..... 293, 301, o462, K81, I161  
 \IndentBox ..... 291, 293, n49, n129  
 \index ..... 773  
 \index N182, P6, P18, R24, R32, V620, V679  
 \indexentry ..... P15  
 \inf ..... H25  
 \infty ..... A316  
 \initcatcodetable ..... d91  
 \input ..... 324, 789  
 \input ..... 182,  
     196, 786, 835, 836, 840, 842, 848,  
     a68, a174, a177, a234, d18, e108,  
     e130, f22, q544, w16, x106, z751,  
     z761, z771, A10, A11, A12, A13,  
     A14, A23, A41, A42, A46, A47,  
     A136, A137, A138, A139, A654,  
     A655, A656, D1129, S615, X150,  
     X164, X189, X267, X311, X391, X579  
 \input@path ..... 1, 6  
 input@path commands:  
     \input@path: ..... a239  
 \inputencodingname ... X368, X390, X472  
 \InputIfFileExists ..... 324, 789  
 \InputIfFileExists ..... 340, 342,  
     812, 813, 842, 844, 855, q527, q550,  
     q565, q575, q585, q639, r1538,

u391, z743, z753, z763, D845,  
 D1212, S910, S976, T143, W8, X261  
 \inputlineno ..... a327, l214  
 \insert ..... 289, b254, b279, b281, b284, b299,  
 O465, O484, O502, V517, V518, V1883  
 \int ..... A348  
 int commands:  
     \int\_add:Nn ..... g520, g1104  
     \int\_compare:nNnTF ..... g388, g515,  
     g770, g812, g1692, g1995, g1997,  
     h791, h813, h823, h1305, i161, i292,  
     y194, U55, U111, U141, U351, U357  
     \int\_decr:N .....  
         ..... g467, g481, g495, g1691, h802  
     \int\_eval:n .....  
         ..... g2009, h810, h863, h870, U353  
     \int\_gdecr:N ..... T522  
     \int\_gincr:N ..... g967, T512, U87, U102  
     \int\_gset:Nn ..... y199, T508  
     \int\_incr:N ..... 135, g292, g411, g784,  
         g849, g856, g1071, g1302, g1701, h770  
     \int\_new:N ..... g15, g27,  
         g35, g1671, h759, i7, T507, U344, U346  
     \int\_set:Nn ..... g989, g999, g1168,  
         g1177, g1227, g1675, g2102, g2128,  
         i156, i199, i342, i343, T212, T226  
     \int\_step\_inline:nnn ..... i164, i178  
     \int\_use:N ..... 862,  
         g25, g815, g971, g999, g1089, g1093,  
         g1094, g1177, g1293, y182, y183,  
         y185, y194, T516, U104, U116, U361  
     \int\_value:w ..... U48, U136  
     \int\_zero:N ..... g283,  
         g377, g690, g693, g819, g1063, h766  
     \c\_max\_int ..... U212, U238, U263, U290  
     \c\_zero\_int ..... i161, U91  
 \interdisplaylinepenalty .....  
     ..... o13, H55, H194, H380  
 \interfootlinepenalty ..... b394  
 \interfootnotelinepenalty .....  
     ..... b394, o18, O467, O486, O504  
 \interlinepenalty .. o11, u663, G426,  
     G429, G447, G450, N67, N118,  
     N209, N232, O467, O486, O504,  
     V338, V1153, V1157, V1319, V1323  
 \intextsep ..... V1136, V1140,  
     V1155, V1158, V1165, V1298,  
     V1304, V1321, V1324, V1333, V2337  
 \intop ..... A347, A348  
 \iota ..... A276  
 iow commands:  
     \iow\_char:N .....  
         ... g399, g463, g548, g617, g629,

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltruatex.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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

g660, g1009, g1015, g1676, g2411,  
 g2501, h893, h904, h1320, h1350,  
 h1362, h1367, h1372, h1519, h1521,  
 i397, n111, n118, n119, n120, n121  
 $\backslash\text{iow\_log:n}$  ..... h936  
 $\backslash\text{iow\_newline:}$  g1193, g1203, g1300, T520  
 $\backslash\text{iow\_now:Nn}$  ..... U360  
 $\backslash\text{iow\_term:n}$  .....  
     . g1191, g1201, 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, T513, T546  
 $\backslash\text{isshortstack}$  ..... L132  
 $\backslash\text{itdefault}$  ..... v697, z30, A94  
 $\backslash\text{item}$  288, 646, l280, G351, G401, G403,  
     G417, G439, H439, H451, H478,  
     K78, M36, M38, Q4, Q8, I141, I219  
 $\backslash\text{itemindent}$  645, 646, I9, I42, I95, I187, I208  
 $\text{itemize}(\text{environment})$  ..... I242  
 $\backslash\text{itemize}$  ..... I242  
 $\backslash\text{itemsep}$  ..... 646, I1, I176  
 $\backslash\text{iterate}$  ..... a81, a82, b424  
 $\backslash\text{itshape}$  ..... 447, 527, r447, r807,  
     v695, v696, z28, z29, z554, z587,  
     z593, C21, D634, M36, M38, O399

**J**

$\backslash\text{J}$  ..... X243, X543  
 $\backslash\text{j}$  ..... r248, r403,  
     r548, r779, r1142, r1357, r1443, X552  
 $\backslash\text{jmath}$  ..... A310  
 $\backslash\text{jobname}$  ..... 820, 845  
 $\backslash\text{Join}$  ..... z725  
 $\backslash\text{joinrel}$  A471, A478, A480, A482, A484,  
     A486, A488, A490, A492, A496, A498  
 $\backslash\text{jot}$  ..... H53, H191, H391, H401

**K**

$\backslash\text{k}$  ... 370, r485, r590, r595, r617, r622,  
     r698, r699, r757, r758, r812, r814,  
     r819, r821, r1243, r1311, r1312,  
     r1329, r1330, r1352, r1353, r1354,  
     r1407, r1408, r1441, r1442, D181, D204  
 $\backslash\text{kanjiskip}$  ..... e74  
 $\backslash\text{kappa}$  ..... A277  
 $\backslash\text{ker}$  ..... H27  
 $\backslash\text{kern}$  ..... 874  
 kernel internal commands:  
      $\backslash\text{__kernel_chk_if_free_cs:N}$  .....  
         . 161, 175, g2065, g2066, g2643  
      $\backslash\text{__kernel_cmd_if_xparse:NTF}$  .....  
         . 132, 133, 138,  
         257, g1019, g1224, g2058, g2160, i127

**L**

$\backslash\text{__kernel_exp_not:w}$  .....  
     . h41, h47, h49, h56, h61  
 $\backslash\text{l\_kernel_expl\_bool}$  ..... i297  
 $\backslash\text{__kernel_file_name_sanitize:n}$  . T98  
 $\backslash\text{kerneltmpDoNotUse}$  ..... 265, 266,  
     i335, i347, i353, i358, i364, i371, i384  
 $\backslash\text{kill}$  ..... K154, K162

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

\ldotp .. A501, A504, A619  
 \ldots .. r322, A505  
 \le .. A416, A418  
 \leaders .. b480, A340, A558, A559, A561, A562, K376, L565, L578, L590, L600, N214, N237  
 \leadsto .. z728  
 \leavevmode .. b439, b466, b469, b480, b482, o407, o421, r75, r184, r289, r290, r393, r425, r429, r432, r479, r763, r796, C123, D98, D927, G426, G447, G460, G471, G479, G558, G568, G581, H439, H451, H478, J8, J17, J24, J156, J158, J174, J202, J263, J339, J438, J455, J462, K178, L134, L314, L373, N40, N210, N222, N233, O530, Q14, V157, V162, V184, V189, I58, I103  
 \left .. A625, A627, A629, A631, A636, A637, A638, A639, H154, H160, H182  
 \Leftarrow .. A410, A492, A498  
 \leftarrow .. A437, A439, A480, A490, A496, A550  
 \leftarrowfill .. A534, A550  
 \lefteqn .. H415  
 \leftharpoondown .. A453, A467  
 \leftharpoonup .. A452  
 \lefthyphenmin .. W11  
 \leftline .. J489  
 \leftmargin .. . 644, 646, I9, I52, I53, I94, I146, I148  
 \leftmargini .. 646, H431, I17  
 \leftmarginii .. I17  
 \leftmarginiii .. I17  
 \leftmarginiv .. I17  
 \leftmarginv .. I17  
 \leftmarginvi .. 646, I17  
 \leftmark .. R48  
 \Leftrightarrow .. A409  
 \leftrightarrow .. A436  
 \leftskip .. b461, u658, G359, G365, G369, G379, G383, G387, G419, G441, J295, J316, N207, N212, N230, N235, I74  
 \legacyoldstylenums .. D4, D617  
 \leq .. A414, A416  
 \let .. 86, 89, 246, 402, 550, 554, 855, 881  
 \LetLtxMacro .. 93  
 \lfloor .. A606  
 \lg .. H4  
 \lgrou p .. A608  
 \lhd .. z731  
 \lhook .. A477, A478  
 \lim .. H6  
 \liminf .. H8  
 \limits .. A539, A543, H149, H340  
 \limsup .. H7  
 \line .. I269, L158, L450, L809, L826  
 \linebreak .. 304  
 \linebreak .. o9, o26  
 \linepenalty .. b320  
 \lineskip .. b402, b434, b469, A458, H187, J297, J317, K71, K198, L136, L315, L374, U221, U272, V622, V681  
 \lineskiplimit .. b403, b434, b471, b472, A458, A510, H189, H193, J283, J298, J305, U222, U273, V622, V681  
 \linespread .. u316  
 \linethickness .. L130, L810, L827  
 \linewidth .. 645, q30, q99, q157, H285, H311, H440, H452, H479, H483, H502, J293, J314, K36, O266, V146, V205, I15, I51, I52, I54  
 list (environment) .. I34  
 \list .. I34, I236, I247  
 \listfiles .. 789  
 \listfiles .. 197, 198, q654  
 \listparindent .. 646, I9, I41, I50  
 \ll .. A434  
 \llap .. J493, I238, I249  
 \lmoustache .. A563  
 \ln .. H5  
 \lnot .. A326, A327  
 \LoadClass .. 787  
 \LoadClass .. S621, S651, S875, S993, S1067, S1075, S1076  
 \LoadClassWithOptions .. 787  
 \LoadClassWithOptions .. S650  
 \LoadFontDefinitionFile .. u418, u444, u445, A21, A27, A28, A29, A33  
 \LoadPackageWithOptions .. 836  
 \locount .. d17  
 \log .. H3  
 \loggingall .. b491  
 \loggingoutput .. b487, b505, b523, b539, b553  
 \LogHook .. 185, h1442  
 \long .. 85  
 \Longleftarrow .. A492  
 \longleftarrow .. A489  
 \Longleftrightarrow .. A498, A500  
 \longleftarrow .. A496  
 \longmapsto .. A494  
 \Longrightarrow .. A486  
 \longrightarrow .. A487, A494  
 \loop .. a81, b424, d150, d159, u672, K382, S1116, S1177, S1247, S1310, S1361, S1399, X351, X362, X372, X383, X413, X439, X449

**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=ltCMdHOOKS.dtx, j=ltALLOC.dtx, k=ltCNTRL.dtx, l=ltERROR.dtx, m=ltPAR.dtx, n=ltPARA.dtx, o=ltSPACE.dtx, p=ltLOGOS.dtx, q=ltFILES.dtx, r=ltOUTENC.dtx, s=ltCOUNTS.dtx, t=ltLENGTH.dtx, u=ltFSSbas.dtx, v=ltFSSaxes.dtx, w=ltFSStrc.dtx, x=ltFSScmp.dtx, y=ltFSSdcl.dtx, z=ltFSSini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltFNTcmd.dtx, D=ltTEXTcomp.dtx, E=ltPAGENO.dtx, F=ltXREF.dtx, G=ltMISCEN.dtx, H=ltMATH.dtx, I=ltLISTS.dtx, J=ltBOXES.dtx, K=ltTAB.dtx, L=ltPICTUR.dtx, M=ltTHM.dtx, N=ltSECT.dtx, O=ltFLOAT.dtx, P=ltIDXGLO.dtx, Q=ltBIBL.dtx, R=ltPAGE.dtx, S=ltCLASS.dtx, T=ltFILEHOOK.dtx, U=ltSHIPOUT.dtx, V=ltOUTPUT.dtx, W=ltHYPHEN.dtx, X=ltFINAL.dtx

\looseness . . . . .	b333	\markright . . . . .	R22, R44
\lor . . . . .	A369, A371	\marks . . . . .	d37, X10, X12
\lower . p2, A458, J214, L35, L45, L196, L305, L306, L353, L354, L409, L410		math (environment) . . . . .	H332
\lowercase l26, r141, r1057, r1536, u332, u390, G511, G526, G541, G581, X567		\math . . . . .	H332
\lq . . . . .	b406	\mathaccent . . . . .	y713, y761, y795, y805
lrbox (environment) . . . . .	660	\mathalpha . . . . .	y883, y1062, A169, A170, A171, A172, A173, A174, A175, A176, A177, A178, A179, A180, A181, A182, A183, A184, A185, A186, A187, A188, A189, A190, A191, A192, A193, A194, A195, A196, A197, A198, A199, A200, A201, A202, A203, A204, A205, A206, A207, A208, A209, A210, A211, A212, A213, A214, A215, A216, A217, A218, A219, A220, A221, A222, A223, A224, A225, A226, A227, A228, A229, A230, A297, A298, A299, A300, A301, A302, A303, A304, A305, A306, A307, A516, A517, A518, A519, A520, A521, A522, A523, A525, A528
\lua commands:		\mathbf . . . . .	
\lua_now:n . . . . .	U31	z14, z255, z308, z353, z381, z475, A151	
\luabytecode . . . . .	d194	\mathbin y1067, A232, A233, A235, A358,	
\luachunk . . . . .	d202	A359, A360, A361, A364, A365, A366, A367, A370, A371, A372, A373, A374, A375, A376, A377, A378, A379, A380, A381, A382, A383, A384, A385, A386, A387, A388, A389, A390, A391, A392, A393, A394, A395, A396, A397, H37	
\luadef . . . . .	U42	\mathcal . . . . .	A150
\luafunction . . . . .	41, d178	\mathchar . . . . .	
\luatexbase . . . . .	d280	.b467, y824, y868, A335, A336, A617	
\luatexluafunction . . . . .	a18, a23	\mathchardef . b21, b22, b23, b24, b107, b110, b111, d219, j3, j4, j5, j6, r70, y859	
\luatexversion . . . . .	a11, d5, e73, e119, r994	\mathcharzero . . . . .	d219
		\mathchoice . . . . .	H61
		\mathclose . . . . .	y1070, A231,
		A240, A242, A245, A250, A256, A258, A260, A566, A593, A597, A601, A605, A611, H43, H46, H49, H52	
M		\mathcode . . . . .	y856, A252, A253, A254
\M . . . . .	b404	\mathdollar . . . . .	r307, A614
\magstep . . . . .	b395	\mathellipsis . . . . .	r321, A619
\magstephalf . . . . .	b395	\mathgroup . . . . .	
\makeatletter . . . . .		b79, u14, w303, w309, w315, w316, w327, A643, D8, D14, D614, D1139	
f728, i293, q32, q101, q159, u396, G26, G79, N151, S615, S839, S954, V2		\mathhexbox . . . . .	b467, z652
\makeatother . . . . .	f728, i293, S615, X635	\mathindent . . . . .	H429, H441, H453, H481, H492
\makebox . . . . .	660	\mathinner . . . . .	A504, A508, A513, A619
\makebox . . . . .	H285, H311, J3	\mathit . . . . .	v696, z29, A153, A156, A617
\makeglossary . . . . .	773	\mathnormal . . . . .	A149
\makeglossary . . . . .	q204, P20		
\makeindex . . . . .	773		
\makeindex . . . . .	q203, P3		
\makelabel . . . . .			
.. 645, I45, I97, I205, I218, J238, I249			
\MakeLowercase . . . . .	X561, X573		
\MakeRobust . . . . .	504, f271, f773, f774, f775, f776, f777, f778, f779, f780, f781, f782, f783, f784, f785, f786, f787, f788, y796, y991, L806, L807, L808, L809, L810, L811, L812, L813, L814, L815, L816, L817		
\maketitle . . . . .	737		
\MakeUppercase . . . . .	F48, F60, X551		
\mapsto . . . . .	A444		
\mapstochar . . . . .	A443, A444, A494		
\marginpar . . . . .	291, O308		
\marginparpush . . . . .	V85, V1834		
\marginparsep . . . . .	V84, V1845, V1847		
\marginparwidth . O341, O359, V83, V1847			
\mark . . . . .	R27, R35, R53		
\markboth . . . . .	R21, R22, R41, R43		

**File Key:** a=ltirchk.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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfsstrc.dtx, x=ltfscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=lpageno.dtx, F=lxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

\mathop . . . . . y1066, A341, A342,  
     A343, A344, A345, A346, A347,  
     A349, A350, A351, A352, A353,  
     A354, A356, A357, A537, A540, H3,  
     H4, H5, H6, H7, H8, H9, H10, H11,  
     H12, H13, H14, H15, H16, H17,  
     H18, H19, H20, H21, H22, H23,  
     H24, H25, H26, H27, H28, H29,  
     H30, H31, H32, H33, H34, H149, H340  
 \mathopen . . . . .  
     . y1069, A241, A244, A249, A255,  
     A257, A259, A564, A595, A599,  
     A603, A607, A609, H41, H44, H47, H50  
 \mathord . . . . . y883, y1065, A236,  
     A243, A246, A251, A263, A264,  
     A265, A267, A268, A269, A270,  
     A271, A272, A273, A274, A275,  
     A276, A277, A278, A279, A280,  
     A281, A282, A283, A284, A285,  
     A286, A287, A288, A289, A290,  
     A291, A292, A293, A294, A295,  
     A296, A308, A309, A310, A311,  
     A312, A313, A314, A315, A316,  
     A317, A318, A319, A320, A321,  
     A322, A323, A324, A325, A327,  
     A328, A329, A330, A331, A332,  
     A333, A334, A524, A526, A527,  
     A549, A550, A553, A554, A555,  
     A556, A568, A570, A572, A575,  
     A577, A591, A613, A614, A615, A616  
 \mathpalette . . . . . A457,  
     A461, A464, H60, H69, H99, H129  
 \mathparagraph . . . r310, s168, s180, A614  
 \mathpunct . . . . .  
     y1071, A234, A238, A501, A502, A503  
 \mathrel . . . . . y1068, A237, A239,  
     A247, A248, A261, A262, A338,  
     A398, A399, A400, A401, A402,  
     A403, A404, A405, A406, A407,  
     A408, A409, A410, A411, A414,  
     A415, A418, A419, A420, A421,  
     A422, A423, A424, A425, A426,  
     A427, A428, A429, A430, A432,  
     A433, A434, A435, A436, A437,  
     A438, A441, A442, A443, A445,  
     A446, A447, A448, A449, A450,  
     A451, A452, A453, A454, A455,  
     A457, A461, A464, A471, A473,  
     A476, A477, A479, A482, A484,  
     A579, A581, A583, A585, A587,  
     A589, H42, H45, H48, H51, H149, H340  
 \mathring . . . . . A528  
 \mathrm . . . . . z5, z406, z449, z481, A148  
 \mathsection . . . . . r311, s167, s179, A614  
 \mathsf . . . . . z8, z411, z454, z484, A152, A155  
 \mathsterling . . . . . r319, A614  
 \mathstrut . . . . . H84, H93, H158, H159  
 \mathsurround . . . . . b455  
 \mathsymbol . . . . . y861  
 \mathtt . . . . . z11, z416, z459, z487, A154  
 \mathunderscore . . . . . A614  
 \mathversion . . . . . u336, z614, z616  
 \matrix . . . . . H156, H160, H167  
 \max . . . . . H22  
 \maxdeadcycles . . . . . V7  
 \maxdepth . . . . . b376, o287, q60, q128, q183,  
     V92, V169, V170, V506, V514,  
     V546, V715, V724, V764, V991, X138  
 \maxdimen . . . . . 880, b309,  
     b377, b378, b434, b472, b488, b503,  
     b504, b522, b538, b553, u645, u655,  
     u690, u705, w384, w437, A458,  
     L475, L503, L533, L611, L628,  
     S1471, S1512, S1521, U349, U351,  
     U398, V291, V1853, V1873, V1878,  
     V2199, V2239, V2240, V2242, X142  
 \mbox . . . . . 660  
 \mbox . . . . . b467, p13, r293,  
     r409, r568, r1157, z648, A506, J11,  
     J20, J24, L52, O409, O416, O437, O444  
 \mddefault . . . . . 510, 511, 514, 516,  
     z18, z284, z290, z291, z292, z331,  
     z332, z333, z342, z369, z385, z402,  
     z443, z479, A92, A105, A107, A121  
 \mdseries . . . . . 199, 517, 525, z16,  
     z17, z221, z277, z328, z329, z363,  
     z364, z383, z384, z477, z478, z651, C20  
 mdseries . . . . . z420  
 mdseries/defaults . . . . . z420  
 \meaning . . . . .  
     a219, a228, a323, f228, f290, f328,  
     f356, f439, f699, y535, y548, y649,  
     y714, y761, y825, y919, y1015, y1119  
 \medbreak . . . . . b449, f780, f801  
 \mediumseries . . . . . 517  
 \medmuskip . . . . .  
     . A645, H36, H38, H211, H214, H228  
 \medskip . . . . . b452, o400  
 \medskipamount . . . . . b451, o401, o403  
 \medspace . . . . . H201  
 \MessageBreak . . . . . 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, r161, r988, r1541, r1544,  
     u34, u35, u536, u570, v456, w20,  
     w21, w67, w88, w327, w478, w498,  
     w530, w546, w561, w574, x31, x33,

**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=ltspacedt.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

y181, y490, y499, y637, z61, z94, C144, D23, D79, D81, D100, D851, D853, D854, D855, D857, D859, D860, D861, D862, D863, D913, D915, D922, D929, D1144, G43, G83, S290, S304, S677, S688, S690, S692, S703, S827, S828, S830, S831, S832, S834, S836, S870, S871, S872, S873, S961, S962, S964, S965, S966, S968, S970, S988, S989, S990, S991, S1052, S1069, S1070, S1138, S1155, S1194, S1269, S1288, S1327, S1382, S1416, S1525, S1527, S1609, S1612, S1625, S1627, U99, U173, U466, U467, U468, U469, V578, V1994, V2031, X286, X287, X288, X290  
 \mho ..... z724  
 \mid ..... A402  
 \min ..... H23  
 minipage (environment) ..... 661  
 \minipage ..... J326  
 \mit ..... z776  
 \mkern ..... A335, A338, A340, A462, A471, A513, A514, A515, A545, A546, A547, A548, A549, A550, A551, A552, H36, H37, H40, H73, H74, N215, N238  
 mlist commands:  
 \mlist\_to\_hlist ..... d843  
 mode commands:  
 \mode\_if\_horizontal:TF ..... n53, n58  
 \mode\_if\_inner:TF ..... n54  
 \mode\_if\_math:TF ..... y211  
 \mode\_if\_vertical:TF ..... n72, n84  
 \models ..... A484  
 module commands:  
 module\_error ..... d337  
 module\_info ..... d337  
 module\_warning ..... d337  
 \module\_error ..... 44  
 \module\_info ..... 44  
 \module\_warning ..... 44  
 modules ..... d290  
 \month ..... a185, c17, S1172, S1305, S1394  
 \moveright ..... V625, V684  
 \mp ..... A389  
 \mscount ..... K379  
 msg commands:  
 \msg\_error:nn ..... h274, h291, n80, n90  
 \msg\_error:nnn ..... g277, g659, g2048, g2052, g2544, g2547, g2555, h73, h299, h337, h355, h1091, T75  
 \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  
 \msg\_error:nnnnn ..... g603, g1723, g1730, g2008, h278, h309, h318  
 \msg\_error:nnnnnn ..... h657, h876  
 \msg\_expandable\_error:nn ..... h180  
 \msg\_expandable\_error:nnn ..... g348, g2613, h208  
 \msg\_expandable\_error:nnnn ..... g1900, g1927, T431  
 \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 ..... g1007, g2410, g2465, g2470, g2475, g2480, g2485, g2497, h1323, h1328, h1359, h1365, h1370, h1375, h1380, h1384, h1391, h1517, T437  
 \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  
 \msg\_redirect\_module:nnn ..... g49  
 \msg\_warning:nnn ..... g73, h152  
 \msg\_warning:nnnn ..... g640, h618, h630, h1516  
 \msg\_warning:nnnnn ..... h449  
 \msg\_warning:nnnnnn ..... h884  
 \mskip ..... H36, H38, H206, H225, H228, H230, H231  
 \mu ..... A279  
 \mubyte ..... X338  
 \multicolumn ..... 617, K233  
 \multiput ..... L78, L811, L828  
 \multispan ..... K233, K379  
 \muskip ..... b29, b55, b93, d34, A545, A546  
 \muskipdef ..... b55, b93, d220  
 \muskipzero ..... d220

## N

\n ..... d324, d326, d333, d335, d462, d570, d596, d622, d668, d690, d709, d717, d718, d738, d751, d758, d759, d766, d778, X107, X112  
 \nabla ..... A319

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

```

\NAME 74
\narrower b460
\natural A329
\ncallback d673
\ndefault d678, d682
\n 550, A413
\nearrow A405
\NeedsTeXFormat . w12, D819, S682, S1671
\neg A325, A326
\negmedspace H201
\negthickspace H201
\negthinspace o468, H201
\neq 550, A412
new commands:
 new_attribute d403
 new_bytocode d437
 new_chunkname d450
 new_luafunction d466
 new_whatsit d425
\new_attribute 42
\new_bytocode 42
\new_chunkname 42
\new_luafunction 42
\new_whatsit 42
\newattribute 41
\newattribute d74, d230
\newbox b47,
 b314, b457, j13, G482, H66, J115,
 K16, K17, K18, K343, L6, L670,
 L675, V86, V120, V121, V122, I27
\newcatcodetable 41
\newcatcodetable
 d84, d93, d94, d120, d121, d234
\newcommand 74
\newcommand 86, 89, 92, 95, f77,
 r4, v530, v535, v540, z36, z72, A51,
 A52, A53, A54, A56, A57, A59,
 A60, A92, A93, A94, A95, A96,
 A97, A120, A121, A122, G303,
 G304, G305, G306, L682, T536,
 T537, T538, T539, T541, V2325,
 V2328, V2331, V2332, V2335, V2336
\NewCommandCopy
 89, 91, 92, f461, f463, g984, g1160
\newcount b47, b352, b394, j7, j8, o124,
 q7, s36, w25, y27, y142, y377, H55,
 H344, H345, J367, K11, K12, K13,
 K14, K15, K335, K336, K337, L664,
 L665, L666, L667, L676, N36, N140,
 N141, O3, O267, O268, O269,
 O270, S1468, V103, V105, V107,
 V109, V111, V119, V2020, V2323,
 V2326, V2329, V2333, X3, X4, X5,
 X77, I23, I24, I25, I26, I56, I226, I241
\newcounter 391
\newcounter s10
\newdimen
 b47, b309, b311, b312, b393, j10,
 j11, j12, o123, w398, w399, H53,
 J171, J172, K3, K5, K6, K7, K8,
 K166, K338, K339, K340, K341,
 L3, L4, L5, L7, L431, L432, L433,
 L434, L435, L436, L668, L669,
 L671, L672, L673, L674, O451,
 V71, V72, V73, V75, V76, V77,
 V78, V79, V80, V81, V82, V83,
 V84, V85, V91, V93, V94, V106,
 V108, V110, V112, V113, V114,
 V115, V116, V117, V118, V2021,
 V2022, I9, I10, I11, I12, I13, I14,
 I15, I16, I17, I18, I19, I20, I21, I22
\NewDocumentCommand
 89, 103, 162, 253, g2505, h1396,
 h1397, h1398, h1401, h1402, h1410,
 h1412, h1414, h1416, h1418, h1430,
 h1432, h1446, h1448, h1451, s79, s87
\NewDocumentEnvironment
 g2284, g2291, g2541
\newenvironment 75
\newenvironment
 195, f146, S1170, S1303, S1392
\NewExpandableDocumentCommand . g2561
\newfam b47, d38, u16
\newfont z618
\newgroup y47
\newhelp b306
\NewHook 176, 177,
 184, 190, 192, 194, 195, 204, 252,
 253, h1396, h1532, q72, q73, q74,
 q343, w150, z420, z421, z422, z423,
 z424, z425, z426, z427, z428, G33,
 G34, G35, G36, G37, S939, S940, T177
\NewHookPair 251
\newif c70, f168,
 f767, j9, q5, q6, u204, v401, v413,
 y15, z528, C82, D871, F3, H75,
 H76, H190, H346, J423, K19, K251,
 L157, L427, L428, L429, L430,
 L459, L460, N38, N124, S2, V95,
 V96, V97, V98, V99, V100, V101,
 V102, I28, I29, I30, I31, I32, I33, I38
\newinsert
 b193, b242, J368, O390, V27, V1852
\newlabel F22, F34
\newlanguage b47, X270
\newlength 399
\newlength t3
\newline o92, o99, o105

```

**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=ltspacedtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfssrc.dtx, x=ltfscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

```

\newlinechar a72, f20 \nobreakdashes o406
\newluabytocode 41 \nobreakspace o420
\newluabytocode d189, d244 \nobreakspace_ 320
\newluachunkname 42 \nocide 779
\newluachunkname d197, d246 \nocite 776
\newluafunction 41 \nocite Q39
\newluafunction d4, d173, d228, d240, r1010, U28 \nocorr C43, C58, C62, C65
\newmarks X6 \nocorlist C89, C121
\newmathalphabet x13, x109 \nofiles 324
\NewMirroredHookPair .. 177, h1396, h1534 \nofiles 611, 613, 879, q198
\NewModuleRelease c149, g9, h4, i4, n4, D2 \noindent 288, 291,
\newmuskip b47 293, 296, 301, 654, u668, u694, N139
\newpage V133, V139, V150 \nointerlineskip .. b432, A339, A531,
\newread b47, b307 A534, A538, A542, H284, H310,
\NewReversedHook 177, 182, 190, 252, h1396, L563, L566, L576, L578, V1842, V1850
\h1533, q344, q345, S941, S942, T178 \nolimits A348, A355, H3, H4,
\newrobustcmd 89 H5, H9, H10, H11, H12, H13, H14,
\newsavebox 660 H15, H16, H17, H18, H19, H20,
\newsavebox J115 H21, H26, H27, H28, H29, H31, H34
\newskip b47, b310, b313, b391, \nolinebreak 304
b392, j14, j15, j17, o403, o404, o405, \nolinebreak o9, o27
o454, t3, G400, H347, H430, V2337, \nonfrenchspacing
V2338, V2339, V2343, V2344, b398, b643, f782, f803, q48, q117, q175
V2347, V2348, V2349, V2353, \nonscript H36, H38
V2354, V2355, I2, I3, I4, I5, I6, I7, I8 \nonumber H374, H413, H414
\newtheorem M1 \nopagebreak 304
\newtie 576, r851, D188, \nopagebreak 67, o25
D189, D208, D705, D1015, D1016 \noprotrusion N222, N245
\newtoks b47, \normalbaselines b402, H154, H156
b306, j16, n35, u346, u347, v499, w247 \normalbaselineskip
\newwhatsit 41 b391, b403, w188, J299, J318
\newwhatsit d181, d242 \normalcolor 954, H339,
\newwrite b47, b308, q3, q4, N154, P4, P21 H426, J89, J359, N218, N241,
\newXeTeXintercharclass X21 O97, O166, V216, V492, V629,
nfss internal commands: V639, V688, V698, V2260, V2293
 _nfss_init_mv_freeze:N . y186, y210 \normalfont 198, 352, 517, u646,
\NG r531, r1137, X571 u706, z653, z673, z675, z684, z689,
\ng r552, r1138, X571 z691, z699, z706, z708, z714, C18,
\ni A430, A431 G461, H339, H426, N218, N241, O401
\noalign 617, A339, A531, A534, A537, \normalfont z420
A538, A542, A543, H158, H159, \normallineskip .. b391, b402, J297, J317
H175, H178, H192, H391, H401, \normallineskiplimit
K224, K230, K359, K378, L148, L154 b391, b403, H193, J282, J298, J304
\nobreak .. 300, b437, b440, b442, f781, \normalmarginpar O387
f802, o59, o71, o115, o141, o147, \normalsfcodes
o160, o173, o199, o351, o359, o385, q44, q46, q48, q113, q115, q117,
o393, o414, o421, o452, q202, q214, q171, q173, q175, q197, V618, V677
r409, r435, r437, r568, r1157, G325, \normalshape 442, 447, v682, v723
G332, J488, N90, N212, N213, \normalsize 179, q42, q111, q169, C142,
N217, N235, N236, N240, O531, O23, O176, O372, S5, V617, V676
R29, R37, V336, V1149, V1315, \not A338, A412, A413, A435
X194, X196, X200, X201, X202, X206 \notexpanded 262
\ntin A461
\nu A280

```

**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=ltSPACE.dtx, p=ltLOGOS.dtx, q=ltFILES.dtx,  
r=ltOUTENC.dtx, s=ltCOUNTS.dtx, t=ltLENGTH.dtx, u=ltFSSBAS.dtx, v=ltFSSAXES.dtx,  
w=ltFSSTRC.dtx, x=ltFSSCMP.dtx, y=ltFSSDCL.dtx, z=ltFSSINI.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltFNTCMD.dtx, D=ltTEXTCOMP.dtx, E=ltPAGENO.dtx, F=ltXREF.dtx,  
G=ltMISCEN.dtx, H=ltMATH.dtx, I=ltLISTS.dtx, J=ltBOXES.dtx, K=ltTAB.dtx, L=ltPICTUR.dtx,  
M=ltTHM.dtx, N=ltSECT.dtx, O=ltFLOAT.dtx, P=ltIDXGLO.dtx, Q=ltBIBL.dtx, R=ltPAGE.dtx,  
S=ltCLASS.dtx, T=ltFILEHOOK.dtx, U=ltSHIPOUT.dtx, V=ltOUTPUT.dtx, W=ltHYPHEN.dtx,  
X=ltFINAL.dtx

\null . . . . .	<u>b416</u> , r327, r363, r486, r489, r834, r837, r1245, F17, G426, G447, G558, G568, H112, H121, H156, H185, N212, N235, U374		\otimes . . . . .	A386																																																																																																																						
\nulldelimiterspace . . . . .	b380, A642		\outer . . . . .	d21, d38																																																																																																																						
\nullfont . . . . .	G167		\outerparskip . . . . .	I1																																																																																																																						
\number a86, c49, c58, d105, f2, f114, s142, u596, u599, w439, y64, y93, y113, y128, y164, y232, y262, y294, z645, S1083, S1172, S1305, S1394, U349			\output . . . . .	V256																																																																																																																						
\numberline . . . . .	N72, N82, <u>N248</u> , O17		\outputpenalty . . . . .																																																																																																																							
\numexpr . . . . .	b189, b205, b215, b246, d82, d105, d157, f642, r1035, y150, V36		. . . . .	V258, V272, V295, V298, V299, V334, V1159, V1160, V1325, V1328																																																																																																																						
\nunknown . . . . .	d695		\oval . . . . .	L450, <u>L453</u> , L812, L829																																																																																																																						
\narrow . . . . .	A407		\over . . . . .	A469, H149, H341																																																																																																																						
<b>O</b>																																																																																																																										
\O . . . . .	r244, r400, r533, r770, r1122, X570		\overbrace . . . . .	A536																																																																																																																						
\o . . . . .	r253, r405, r554, r781, r1128, X570		\overfullrule . . . . .	b375, R69																																																																																																																						
\oalign . . . . .	<u>b469</u>		\overleftarrow . . . . .	A533																																																																																																																						
\obeycr . . . . .	<u>o480</u>		\overrightarrow . . . . .	A530																																																																																																																						
\obeylines . . . . .	<u>b419</u> , f783, f804, G432, G453, G549, G550, V583		\owns . . . . .	A431, A432																																																																																																																						
\obeyspaces . . . . .	<u>b419</u> , f784, f805, V583		<b>P</b>																																																																																																																							
\oddsidemargin . . . . .	V72, V74, V611, V670		\odot . . . . .	A384		\P . . . . .	r310	\OE . . . . .	r243, r399, r532, r769, r1139, X570		package/. /after . . . . .	837	\oe . . . . .	r252, r404, r553, r782, r1140, X570		package/. /before . . . . .	837	\of . . . . .	H67, H343		package/after . . . . .	837	\offinterlineskip . . . . .	<u>b432</u>		package/before . . . . .	837	\oint . . . . .	A355		\PackageError . . . . .	c74, c132, c144, <u>l84</u> , r1539, D825, D877, D921	\ointop . . . . .	A354, A355		\PackageInfo . . . . .	<u>l84</u> , D829, D846, D851, D867, D868, D928, D1213	\oldstylenums 355, 570, 581, <u>D4</u> , D376, D377, D378, D379, D380, D381, D382, D383, D384, D385, D610, D1136			\PackageNote . . . . .	I136	\Omega . . . . .	A307		\PackageNoteNoLine . . . . .	I136	\omega . . . . .	A290		\PackageWarning . . . . .		\ominus . . . . .	A387		. . . . .	<u>l84</u> , D827, D878, D1142, U465	\omit H178, H179, K369, K372, K379, K383			\PackageWarningNoLine . . . . .	<u>l84</u> , r986, V1993	\OmitIndent . . . . .	291, 293, 298, <u>n49</u> , n128		\pagebreak . . . . .	304	\onecolumn . . . . .	<u>V141</u>		\pagebreak . . . . .	<u>o6</u> , <u>o7</u> , o22, o24	\OnlyDescription . . . . .	w5, B3		\pagegoal . . . . .	V1880, V1887	\ooalign . . . . .	<u>b469</u> , r327, r357, r394, r480, r486, r488, r499, r515, r731, r764, r834, r837, r894, r1245, z650, A462, A465		\pagenumbering . . . . .	602	\openin . . . . .	835		\pageref . . . . .	E5	\openup . . . . .	<u>H186</u> , H191		\pageshrink . . . . .	V538, V542, V558	\oplus . . . . .	A388		\pagestyle . . . . .	R2	\OptionNotUsed . . . . .	<u>S446</u> , S471, S1007		\pagetotal . . . . .	V128	or commands:			\paperheight . . . . .	V93	\or: . . . . .	g1242, g1243, h690, h706		\paperwidth . . . . .	V93	\oslash . . . . .	A385		\par . . . . .	289,	\OT . . . . .	r372			290, 292, 293, 298–302, 612, 654, a120, b11, b412, b420, b421, b436, b445, b446, b447, b449, b451, b453, d156, f9, f21, m3, m4, m5, <u>n98</u> , n134, u671, G109, G165, G325, G332, G424, G445, J288, J309, J355, J385, J402, K195, K385, N41, N90, N220, N242, O15, O24, O249, O344, O478, O496, R62, R63, U390, V166, V193, V257, V1886, I63, I110, I127, I129, I135, I161, I164
\odot . . . . .	A384		\P . . . . .	r310																																																																																																																						
\OE . . . . .	r243, r399, r532, r769, r1139, X570		package/. /after . . . . .	837																																																																																																																						
\oe . . . . .	r252, r404, r553, r782, r1140, X570		package/. /before . . . . .	837																																																																																																																						
\of . . . . .	H67, H343		package/after . . . . .	837																																																																																																																						
\offinterlineskip . . . . .	<u>b432</u>		package/before . . . . .	837																																																																																																																						
\oint . . . . .	A355		\PackageError . . . . .	c74, c132, c144, <u>l84</u> , r1539, D825, D877, D921																																																																																																																						
\ointop . . . . .	A354, A355		\PackageInfo . . . . .	<u>l84</u> , D829, D846, D851, D867, D868, D928, D1213																																																																																																																						
\oldstylenums 355, 570, 581, <u>D4</u> , D376, D377, D378, D379, D380, D381, D382, D383, D384, D385, D610, D1136			\PackageNote . . . . .	I136																																																																																																																						
\Omega . . . . .	A307		\PackageNoteNoLine . . . . .	I136																																																																																																																						
\omega . . . . .	A290		\PackageWarning . . . . .																																																																																																																							
\ominus . . . . .	A387		. . . . .	<u>l84</u> , D827, D878, D1142, U465																																																																																																																						
\omit H178, H179, K369, K372, K379, K383			\PackageWarningNoLine . . . . .	<u>l84</u> , r986, V1993																																																																																																																						
\OmitIndent . . . . .	291, 293, 298, <u>n49</u> , n128		\pagebreak . . . . .	304																																																																																																																						
\onecolumn . . . . .	<u>V141</u>		\pagebreak . . . . .	<u>o6</u> , <u>o7</u> , o22, o24																																																																																																																						
\OnlyDescription . . . . .	w5, B3		\pagegoal . . . . .	V1880, V1887																																																																																																																						
\ooalign . . . . .	<u>b469</u> , r327, r357, r394, r480, r486, r488, r499, r515, r731, r764, r834, r837, r894, r1245, z650, A462, A465		\pagenumbering . . . . .	602																																																																																																																						
\openin . . . . .	835		\pageref . . . . .	E5																																																																																																																						
\openup . . . . .	<u>H186</u> , H191		\pageshrink . . . . .	V538, V542, V558																																																																																																																						
\oplus . . . . .	A388		\pagestyle . . . . .	R2																																																																																																																						
\OptionNotUsed . . . . .	<u>S446</u> , S471, S1007		\pagetotal . . . . .	V128																																																																																																																						
or commands:			\paperheight . . . . .	V93																																																																																																																						
\or: . . . . .	g1242, g1243, h690, h706		\paperwidth . . . . .	V93																																																																																																																						
\oslash . . . . .	A385		\par . . . . .	289,																																																																																																																						
\OT . . . . .	r372			290, 292, 293, 298–302, 612, 654, a120, b11, b412, b420, b421, b436, b445, b446, b447, b449, b451, b453, d156, f9, f21, m3, m4, m5, <u>n98</u> , n134, u671, G109, G165, G325, G332, G424, G445, J288, J309, J355, J385, J402, K195, K385, N41, N90, N220, N242, O15, O24, O249, O344, O478, O496, R62, R63, U390, V166, V193, V257, V1886, I63, I110, I127, I129, I135, I161, I164																																																																																																																						

**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=ltspacedt.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfsstrc.dtx, x=ltfscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

para commands:

- \para\_end: ..... 292, 299, 301, n51, n98, n99, n100
- \g\_para\_indent\_box ..... 293, 296, n14, n42, n44, n47, n49, n75
- \para\_omit\_indent: 293, 298, n46, n50
- \para\_raw\_end: ... 293, 301, n71, n96
- \para\_raw\_indent: . 293, 301, n71, n94
- \para\_raw\_noindent: 293, 301, n71, n95

para internal commands:

- \\_para\_handle\_indent: . n31, n43, n77
- \g\_\_para\_standard\_everypar\_tl ... 297, 298, n12, n34, n36, n76, n87

para/after ..... 291, n6

para/before ..... 291, n6

para/begin ..... 291, n6

para/end ..... 291, n6

\paracntvalue ..... 294

\paragraphmark ..... N143

\parallel ..... A401

\parbox ..... 660

\parbox ..... 289, 291, J234

\parboxrestore ..... J322

\parfillskip ..... 300, b390, u645, u660, u705, G361, G371, G380, G388, G420, G442, J296, J317, N207, N230, I76

\parindent ..... 288, b382, b461, b462, G361, G366, G371, G380, G384, G388, G420, G442, J291, J312, N208, N231, I50

\parsep ..... 646, I1, I49, I90

\parseunicodedataI ..... d123, d162

\parseunicodedataII ..... d124, d126

\parseunicodedataIII ..... d128, d134

\parseunicodedataIV ..... d130, d142

\parseunicodedataV ..... d146, d149

\parshape ..... 296, I54

\parskip ..... 288, 291, 295, 297, 646, b383, G336, G418, G420, G440, G442, H498, J291, J312, K79, V1159, V1327, I49, I73, I88, I90, I117, I153, I172, I223

\partial ..... A315

\partopsep ..... 646, H496, I1, I61

\PassOptionsToClass ..... 787

\PassOptionsToClass ..... S378

\PassOptionsToPackage ..... 787

\PassOptionsToPackage ..... S378

\patterns ..... r205

\pausing ..... b334

\pdffilesize ..... e71, e117

\pdfgentounicode ..... X301, X302, X306, X321, X326, X327, X328

\pdfhorigin ..... U312

\pdftexrevision ..... X307

\pdftexversion ..... X305, X306, X307

\pdfvariable ..... U311, U316

\pdfvorigin ..... U317

peek commands:

- \peek\_meaning:NTF ..... 165, g2212
- \peek\_meaning\_remove:NTF ..... 150, 165, g1619, g1721, g2200
- \peek\_N\_type:TF .. g1625, g1649, g1681
- \peek\_remove\_spaces:n ..... g1617

\penalty ..... b441, b442, b443, b444, b445, b446, b450, b452, b454, o34, o37, o46, o281, o291, o316, o320, C118, G426, G429, G447, G450, H37, H194, H391, H401, K56, O195, O199, O201, O217, O221, O223, Q17, V136, V176, V195, V198, V1157, V1323, I190

\perp ..... A447

\phantom ..... H75

\Phi ..... A305

\phi ..... A287

\Pi ..... A302

\pi ..... A282

picture (environment) ..... L21

\picture ..... L21

\pm ..... A390

\pmatrix ..... H160, H168

\mod ..... H39

\PopDefaultHookLabel ... 181, 182, h1418

\poptabs ..... I256, K142, K161

\poptracing ..... w148, w340

\postdisplaypenalty o12, H438, H450, H476

\pounds ..... r318

\Pr ..... H32

pre commands:

- \pre\_shipout\_filter ..... 862

\prec ..... A421

\preceq ..... A424

\predisplaypenalty b329, H437, H449, H475

\pretolerance .... b316, u647, u662, u707

\prevdepth ..... 316, b432, b436, b437, o287, o288, o349, o354, o383, o388, H192, O196, O198, O218, O220, V167, V169, V172

\PreviousTotalPages ..... 862, 880, U396

prg commands:

- \prg\_break:n ..... g2603, g2605, g2607, g2609, g2610
- \prg\_break\_point: ..... g2611
- \prg\_do\_nothing: . 128, g339, g851, g1562, i190, i304, i422, T132, T134, U164, U165, U170, U179, U183, U319

**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=ltCMdHOOKS.dtx, j=ltALLOC.dtx, k=ltCNTRL.dtx, l=ltERROR.dtx, m=ltPAR.dtx, n=ltPARA.dtx, o=ltSPACE.dtx, p=ltLOGOS.dtx, q=ltFILES.dtx, r=ltOUTENC.dtx, s=ltCOUNTS.dtx, t=ltLENGTH.dtx, u=ltFSSbas.dtx, v=ltFSSaxes.dtx, w=ltFSStrc.dtx, x=ltFSScmp.dtx, y=ltFSSdcl.dtx, z=ltFSSini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltFNTcmd.dtx, D=ltTEXTcomp.dtx, E=ltPAGENO.dtx, F=ltXREF.dtx, G=ltMISCEN.dtx, H=ltMATH.dtx, I=ltLISTS.dtx, J=ltBOXES.dtx, K=ltTAB.dtx, L=ltPICTUR.dtx, M=ltTHM.dtx, N=ltSECT.dtx, O=ltFLOAT.dtx, P=ltIDXGLO.dtx, Q=ltBIBL.dtx, R=ltPAGE.dtx, S=ltCLASS.dtx, T=ltFILEHOOK.dtx, U=ltSHIPOUT.dtx, V=ltOUTPUT.dtx, W=ltHYPHEN.dtx, X=ltFINAL.dtx

```

\prg_new_conditional:Npnn 618, 868, 870,
..... h130, h522, h536, h694,
h1212, h1225, h1241, h1247, h1253,
h1259, h1268, h1276, h1288, i232
\prg_new_protected_conditional:Npnn
..... 418, h470, h498, i106, i345
\prg_replicate:nn g774,
g817, g862, g917, g2015, g2022, T515
\prg_return_false:
.... 216, 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 A253, A317, H243
\ProcessedArgument
..... 160, g320, g324, g331,
g1948, g1949, g1966, g1968, g1990,
g1999, g2005, g2013, g2030, g2625
\ProcessList g2630
\ProcessOptions r1560,
w71, D844, D881, S472, S563, S1071
\ProcessOptions* S472
\prod A349
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
\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
\prop_new:N
.. g45, h30, h31, h95, h633, h634, i16
\prop_put:Nnn g1467, h896, h907
\prop_set_eq:NN h748
\prop_show:N 235
\proto A398
\protect 618, 868, 870,
871, 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, q210,
r26, r32, r51, r55, r209, r217, y598,
y1146, z638, C143, F12, G193,
G203, G238, G241, G256, G266,
K264, N12, N72, N82, N164, N171,
N177, O17, Q5, T274, U83, U95,
U124, U131, U151, V596, V655, X344
\protected 89, 360, 447,
504, f7, f443, o56, o462, r308, r309,
s194, v682, v686, v689, v692, v695,
v698, v701, v704, y988, z573, G126,
G324, H375, H413, K56, K201,
K208, L142, U42, U485, U486, U487
\providecommand f178, h1541,
h1544, r6, r981, D811, D812, V2004
\ProvideCommandCopy 91
\ProvideDocumentCommand g2505
\ProvideDocumentEnvironment g2541
\ProvideExpandableDocumentCommand g2561
\ProvideHook h1487
\ProvideMirroredHookPair h1487
\ProvideReversedHook h1487
provides commands:
provides_module d291
\provides_module 44
\ProvidesClass 787
\ProvidesClass S358
\ProvidesExplPackage T504
\ProvidesFile
a89, A665, A667, A668, A669, S367
\ProvidesPackage 787
\ProvidesPackage
.... 798, 799, w13, D817, D849,
S281, S360, S362, S1672, T533, U480
\ProvideTextCommand r3, r60
\ProvideTextCommandDefault r57
\Psi A306
\psi A289
\PushDefaultHookLabel .. 181, 182, h1418
\pushtabs .. l256, K138, K139, K158, K160
\pushtracing w117, w321
\put 859, 863, 865, 877,
L56, L58, L70, L72, L325, L326,
L327, L328, L336, L338, L351,
L352, L353, L354, L361, L364,
L384, L385, L386, L387, L393,
L395, L407, L408, L409, L410,
L415, L420, L729, L785, L813, L830

```

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltruatex.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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
X=ltfinal.dtx

<b>Q</b>	
\qbezier .....	701
\qbezier .....	<u>L682</u> , L814, L831
\qbeziermax .....	<u>L681</u> , L707, L708, L768
\qqquad .....	<u>o473</u>
\quad .....	<u>o473</u> , H155, H157, H177, N111
quark commands:	
\q_mark	g1152, g1155, g1849, g1853, g1865, g2142, g2150, g2153, g2158
\q_nil .....	153– 155, g514, g1144, g1155, g1253, g1376, g1397, g1401, g1407, g1416, g1418, g1769, g1793, g1802, g1834, g1849, g1853, g1865, g1871, g1897
\quark_if_nil:NTF .....	155, g525, g1855
\quark_if_nil:nTF .....	g1250
\quark_if_recursion_tail_stop:N .....	g710, g1234
\quark_if_recursion_tail_stop:n .....	g410, g578, g1070, g2155, g2156, h260
\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, g1468, g1483, g1485, g1492, g2145, g2151, h266, i312
\q_recursion_tail .....	117, g387, g700, g1066, g1230, g1483, g1487, g2141, g2144, g2150, g2151, h265, h266, i312
\q_stop	g352, g514, g523, g528, g1152, g1154, g1401, g1418, g2006, g2171
quark internal commands:	
\q__cmd .....	.. 114, 153, 155, 156, g336, g340, g886, g1181, g1182, g1200, g1208, <u>g1383</u> , 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 .....	r555, r783, r1168
<b>R</b>	
\quotesinglbase .....	r556, r1165
<b>R</b>	
\r .....	b410, b411, r236, r388, r431, r470, r610, r637, r647, r673, r756, r795, r1237, r1255, r1281, r1403, r1404, D183, D205
\radical .....	y1012, y1015, y1045
\raggedbottom .....	<u>R53</u>
\raggedleft .....	<u>G367</u> , G385, G396, G403
\raggedright .....	<u>G362</u> , G381, G395, G401
\raise .....	r327, r359, r430, r433, r732, r797, r895, r1245, z651, A465, A513, A515, H73, J473, J482, L61, L73, L105, L117, L195, L305, L452, L495, L526, L551, L619, L636, L637, L740, L796
\raisebox .....	661
\raisebox .....	r872, r1214, <u>J450</u>
\rangle .....	A592
\RawIndent .....	293, <u>n94</u> , n130
\RawNoIndent .....	<u>n94</u>
\RawNoindent .....	293, n95, n131
\RawParEnd .....	293, <u>n94</u> , n132
\RawShipout .....	.. 858–861, 864, 866–869, <u>U150</u> , U427
\rbrace .....	r309, A596
\rbrack .....	b408
\rceil .....	A600
\Re .....	A313
\read .....	342
\ ReadonlyShipoutCounter .....	.. 862, 864, 884, <u>U344</u> , U429
\Ref .....	604
\Ref .....	F47, F52, F59, F63, F70
\ref .....	657, <u>F10</u> , F47, F59, O553
\refstepcounter .....	391
\refstepcounter .....	.. 639, F39, <u>F40</u> , F52, F54, F63, F65, H337, H477, M27, N59, O9, I202
\registernumber .....	43
\registernumber .....	<u>d382</u>
\relax .....	84, 96, 289, 438, 617, 814, 848, 855, 881
\Relbar .....	A476, A484, A486, A492
\relbar .....	A473, A488, A490
\relpenalty .....	b324
remove commands:	
remove_from_callback .....	<u>d749</u>
remove_from_callback .....	44
\RemoveFromHook	178, 179, 184, <u>h1416</u> , h1543
\removelastskip ..	b448, b450, b452, b454
\renewcommand .....	74

**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=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
X=ltfinal.dtx

\renewcommand . . . . .  
     91, 511, 518, 542, f124, A66, A68,  
     A70, A71, A73, A75, A77, A78,  
     A84, A86, A88, A89, A103, A104,  
     A105, A113, A114, H425, H445, H466  
 \RenewCommandCopy . . . . . 91, f461, f463  
 \RenewDocumentCommand . . . . . g2505  
 \RenewDocumentEnvironment . g2298, g2541  
 \renewenvironment . . . . . 75  
 \renewenvironment . . . . . f152, H474, H486  
 \RenewExpandableDocumentCommand . . . . .  
     172, g2561  
 \repeat . . . . . a81, a83, b424, d154,  
     d164, u688, K382, S1120, S1181,  
     S1251, S1314, S1365, S1403, X355,  
     X366, X376, X387, X417, X443, X453  
 \requestedLaTeXdate . . . . .  
     S1466, S1499, S1519, S1605  
 \requestedpatchdate . . . . . S1529, S1606  
 \RequirePackage . . . . . 787  
 \RequirePackage . 182, 196, 836, 863, d24,  
     S611, S618, S661, S670, S1067, V2001  
 \RequirePackageWithOptions . . . . . 787  
 \RequirePackageWithOptions . . . . . S653  
 reserved@**a** commands:  
     \reserved@**a**: . . . . .  
         q232, q304, q359, S1097, S1228  
 reserved@**b** commands:  
     \reserved@**b**: . . . . . S230, S247  
 reserved@**c** commands:  
     \reserved@**c**: . . . . . q658  
 \reservedb . . . . . 402  
 \restorecr . . . . . o480  
 \ReverseBoolean . . . . . g2626  
 \reversemarginpar . . . . . O387  
 \rfloor . . . . . A604  
 \rgroup . . . . . A608  
 \rhd . . . . . z733  
 \rho . . . . . A283  
 \rhook . . . . . A479, A480  
 \right . A625, A627, A629, A631, A636,  
     A637, A638, A639, H155, H160, H184  
 \Rightarrow . . . . . A411, A486, A498  
 \rightarrow . . . . . A438,  
     A440, A444, A478, A488, A496, A549  
 \rightarrowfill . . . . . A531, A547  
 \rightharpoondown . . . . . A455  
 \rightharpoonup . . . . . A454, A466  
 \righthyphenmin . . . . . W11  
 \rightleftharpoons . . . . . A464  
 \rightline . . . . . J489  
 \rightmargin . . . . . 646, I9, I40, I51  
 \rightmark . . . . . R48  
 \rightskip . . . . . b462, u659, G359, G363,  
     G369, G379, G382, G387, G419,  
     G441, J295, J316, N207, N230, 175  
 \rlap r430, r433, r797, H415, H426, J493, K81  
 \rmdefault . . . . . 389,  
     516, 522, 523, z6, z208, z398, z407,  
     z439, z450, z482, A50, A120, D7, D613  
 \rmfamily . . . . . 198, 510, 515, 526, z4,  
     z5, z405, z448, z449, z480, z481, C15  
 rmfamily . . . . . z420  
 \rmoustache . . . . . A565  
 \rmsubstdefault A18, A30, D28, D39, D85  
 \Roman . . . . . 391  
 \Roman . . . . . 862, s138  
 \roman . . . . . 391  
 \roman . . . . . s137  
 \romannumeral . . . . . 618, s143,  
     s144, G192, G209, G255, I43, I234, I245  
 \root . . . . . H66, H343  
 \rootbox . . . . . H66  
 \rq . . . . . b406  
 \rule . . . . . 661  
 \rule . . . . . J384,  
     J401, J418, J424, O477, O495, O513

**S**

\S . . . . . r311  
 \samepage . . . . . 304  
 \samepage . . . . . o11, o28  
 \savebox . . . . . 660  
 \savebox . . . . . J116  
 \savecatcodetable . . . . d117, d168, d170  
 \sb . . . . . H199  
 \sbox . . . . . 660  
 \sbox . . . . . b456, p4, r498,  
     r514, J122, J129, J133, J138, J143, I205  
 scan commands:  
     \scan\_new:N . . . . . h38  
     \scan\_stop: . . . . . 110,  
         298, g1615, g1712, g2171, h375,  
         h382, h411, h419, h451, h463, h471,  
         h499, h1289, h1290, i347, i348, i364,  
         i365, n52, U44, U323, U325, U326  
 scan internal commands:  
     \s\_file\_stop . . . . .  
         T103, T105, T108, T110, T112, T125  
     \s\_hook\_mark . . . . .  
         201, 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,

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

h1278, i23, i25, i34, i36, i112, i117, i233, i244, i352, i355, i369, i381  
 \scdefault ..... v694, z27, A94  
 \scriptfont ..... w338  
 \scriptscriptfont ..... w339  
 \scriptscriptstyle ..... H65, H68  
 \scriptspace ..... b381  
 \scriptstyle ..... A337, H64  
 \scshape .. r300, v692, v693, z25, z26, C23  
 \searrow ..... A406  
 \sec ..... H20  
 \secdef ..... N142  
 \secondoftwo ..... 338  
 \sectionmark ..... N143  
 \selectfont ..... 198, 199, 437, 439, 440, 446, 454, 455, 523, 532, 534, p7, r302, r329, r360, r449, r809, r871, r1213, r1247, r1561, u291, u301, u311, v528, v533, v538, v684, v688, v691, v694, v697, v700, v703, v706, w114, w115, w159, w162, w164, z6, z9, z12, z15, z18, z21, z24, z27, z30, z264, z287, z325, z345, z360, z371, z382, z385, z409, z414, z419, z452, z457, z462, z476, z479, z482, z485, z488, z564, z641, z665, z682, z697, D36, D94, D104, D636, D673, D916, D933, O403, O424  
 selectfont ..... w150  
 seq commands:  
   \seq\_clear:N ..... h765  
   \seq\_clear\_new:N ..... h773  
   \seq\_gpop:NNTF ..... h289, h297, T72  
   \seq\_gpop\_right:NN ..... h268  
   \seq\_gpush:Nn ..... h282, T64  
   \seq\_gput\_right:Nn .. h83, h257, h261  
   \seq\_if\_empty:NTF ..... h256, h307  
   \seq\_if\_exist:NTF ..... T60  
   \seq\_map\_inline:Nn ..... g1967, h713, h789, h807  
   \seq\_mapthread\_function:NNN ... 163  
   \seq\_new:N . g1951, h28, h33, h758, T61  
   \seq\_put\_right:Nn .. h771, h864, h871  
   \seq\_set\_split:Nnn ..... g1965  
   \seq\_use:Nnnn ..... h924, h929  
 \seriesdefault ..... 516, 517, 535, r1562, y349, z220, z222, z660, z678, z694, z711, z773, A120  
 \setattribute ..... 42  
 \setattribute ..... d82, d231  
 \setcounter ..... 391  
 \setcounter .. 484, q390, s2, s37, y143, V2324, V2327, V2330, V2334, I225  
 \SetDefaultHookLabel 181, 182, 210, h1418  
   \setlength ..... 399  
   \setlength ..... 309, 318, o83, o243, o442, t4, H494, H499, H500, H501, J43, J204, J265, J268, J341, J440, J441, J442, J471, J472, J479, J480, J481, K176, K384, V2340, V2341, V2342, V2345, V2346, V2350, V2351, V2352, V2356, V2357, V2358  
 \SetMathAlphabet .....  
   .. u11, x140, x141, y603, A155, A156  
 \setminus ..... A393  
 \setraregcatcode .. d96, d104, d113, d114  
 \SetSymbolFont .. y458, A145, A146, A147  
 \settodepth ..... 399  
 \settodepth ..... t17  
 \settoheight ..... 399  
 \settoheight ..... t17  
 \settowidth ..... 399  
 \settowidth ..... t17  
 \sfcode . b398, b399, b400, b401, b485, o416, q45, q114, q172, X236, X536  
 \sfdefault ..... z9, z212, z399, z412, z440, z455, z485, A50  
 \sffamily ..... 198, 510, 523, 526, z7, z8, z410, z453, z454, z483, z484, C16  
 \sffamily ..... z420  
 \sfsubstdefault ..... A19, A31, D30, D87  
 \shapedefault .....  
   .. 442, 517, 528, 543, r1562, v684, y350, z661, z679, z695, z712, A120  
 \sharp ..... A330  
 \shipout ..... 198, 858– 861, 864, 866, 868, 870, 871, 881, 884, U4, U52, U420, U423, V602, V660  
 shipout commands:  
   \l\_shipout\_box ..... 859, 867, 869, 870, 872, 874, U23, U35, U37, U50, U61, U125, U148, U177, U182, U205, U206, U213, U224, U227, U228, U232, U239, U249, U257, U264, U274, U280, U281, U284, U291, U293, U294, U300, U302  
   \l\_shipout\_box\_dp\_dim ..... 859, U192, U195, U197, U228, U281, U501  
   \l\_shipout\_box\_ht\_dim ..... 859, U191, U195, U197, U227, U247, U280, U500  
   \l\_shipout\_box\_ht\_plus\_dp\_dim ..... 859, U194, U197, U213, U264, U275, U277  
   \l\_shipout\_box\_wd\_dim ..... 859, U193, U197, U239, U291, U502  
   \shipout\_debug\_off: .. 863, U7, U404  
   \shipout\_debug\_on: .. 863, U7, U403  
   \shipout\_discard: .. 861, U341, U400

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

\g\_shipout\_READONLY\_int .....  
       ..... 862, 868, U102, U104,  
       U111, U116, U344, U353, U357, U361  
\g\_shipout\_TOTALPAGE\_int ..... 862  
\g\_shipout\_TOTALPAGES\_int .....  
       ..... 862, 868, U87, U346  
shipout internal commands:  
\\_\_shipout\_ADD\_BACKGROUND\_box:n ..  
       ..... U178, U204, U336, U409  
\\_\_shipout\_ADD\_BACKGROUND\_picture:n ..  
       ..... U69, U335, U413  
\\_\_shipout\_ADD\_FIRSTPAGE\_MATERIAL:Nn  
       ..... U171, U187, U402, U407  
\\_\_shipout\_ADD\_FIRSTPAGE\_SPECIALS:  
       ... 869, 871, 872, U110, U164, U176  
\\_\_shipout\_ADD\_FOREGROUND\_box:n ..  
       ..... U117, U255, U339, U411  
\\_\_shipout\_ADD\_FOREGROUND\_PICTURE:n ..  
       ..... U64, U338, U415  
\\_\_shipout\_DEBUG:n ..  
       ..... 865, U7, U103, U115, U147  
\g\_\_shipout\_DEBUG\_BOOL ..  
       ..... U6, U10, U15, U21  
\\_\_shipout\_DEBUG\_GSET: ..... U7  
\g\_\_shipout\_DISCARD\_BOOL ..  
       ..... U81, U88, U90, U201, U342  
\\_\_shipout\_DROP\_FIRSTPAGE\_SPECIALS:  
       ... 871, 872, U126, U165, U176  
\\_\_shipout\_EXCUSE\_EXTRA\_PAGE: ...  
       ..... U373, U381  
\\_\_shipout\_EXECUTE: ... 867, U46, U52  
\\_\_shipout\_EXECUTE\_CONT: ... U57, U59  
\\_\_shipout\_EXECUTE\_MAIN\_CONT:Nnn  
       ..... 867, 870, U60, U77, U146  
\\_\_shipout\_EXECUTE\_NOHOOKS\_CONT:  
       ..... U143, U145  
\\_\_shipout\_EXECUTE\_RAW: ...  
       ..... 870, U134, U150  
\\_\_shipout\_EXECUTE\_TEST\_LEVEL: ...  
       ..... U49, U54  
\\_\_shipout\_EXECUTE\_TEST\_LEVEL\_RAW: ...  
       ..... U134  
\\_\_shipout\_FINALIZE\_BOX: . U26, U123  
\l\_\_shipout\_FIRSTPAGE\_BOX ..  
       ..... 872, U168, U178, U185  
\\_\_shipout\_GET\_BOX\_SIZE:N ..  
       ..... 873, U85, U107, U190, U205  
\l\_\_shipout\_GROUP\_LEVEL\_TL ..  
       ..... U47, U53, U56, U135, U142  
\c\_\_shipout\_HORIGIN\_TL .. U308, U323  
\\_\_shipout\_INIT\_PAGE\_ORIGINS: ...  
       ..... U308, U322  
\g\_\_shipout\_LASTPAGE\_HANDLED\_BOOL ..  
       ..... U121, U186, U363  
\\_\_shipout\_PICTURE\_OVERLAY:N ..  
       ..... U321, U336, U339  
\l\_\_shipout\_RAW\_BOX .. 869,  
       ..... U25, U138, U146, U148, U177, U182  
\\_\_shipout\_RUN\_FIRSTPAGE\_HOOK: ...  
       ..... 869, 871, 872, U108, U161  
\l\_\_shipout\_SAVED\_BADNESS\_TL ..  
       ..... U202, U208,  
       ..... U216, U229, U234, U242, U251,  
       ..... U259, U267, U279, U286, U296, U304  
\\_\_shipout\_SAVED\_PROTECT: ...  
       ..... U83, U131, U151  
\l\_\_shipout\_TMP\_BOX ..  
       ..... U202, U215, U217, U218,  
       ..... U219, U223, U241, U243, U244,  
       ..... U245, U248, U266, U268, U269,  
       ..... U270, U276, U295, U297, U298,  
       ..... U299, U301, U327, U329, U330, U331  
\c\_\_shipout\_VORIGIN\_TL .. U308, U325  
shipout/after ..... 859, U152  
shipout/background ..... 859, U152  
shipout/before ..... 859, U152  
shipout/firstpage ..... 859, U152  
shipout/foreground ..... 859, U152  
shipout/lastpage ..... 859, U152  
\ShipoutBox ..  
       ..... 858–860, 864, 872, U23, U428, U483  
\ShipoutBoxDepth .. U448, U500  
\ShipoutBoxHeight .. 884, U447, U500  
\ShipoutBoxWidth .. U449, U500  
\shortstack . L132, L147, L152, L815, L832  
\show ..... 92, 94, 95, f505, f575, f576  
show commands:  
\show\_hook:n ..... 232  
\showbox ..... V1915  
\showboxbreadth ..... b370,  
       ..... b488, b565, b589, b606, b631, V1915  
\showboxdepth . b371, b488, b564, b588,  
       ..... b605, b632, u647, u691, u708, V1915  
\ShowCommand 89, 92, 94, f498, g1163, g1307  
\ShowDocumentCommandArgSpec ..... g2631  
\ShowDocumentEnvironmentArgSpec .. g2631  
\ShowFloat ..... V1896  
\ShowHook ... 185, 189, 190, h1442, h1548  
\showhyphens ..... u636  
\showoutput ..... b487  
\showoverfull b486, b489, b511, b546, b554  
\showtokens ..... 96, f611  
\Sigma ..... A303  
\sigma ..... A284  
\sim ..... A445, A457  
\simeq ..... A446

**File Key:** a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltnuataex.dtx, e=ltexpl.dtx,  
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdbhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
X=ltfinal.dtx

\sin .....	H9	\stepcounter .....	391
\sinh .....	H11	\stepcounter .....	s17, s27,
\skew .....	A544	u601, y48, F41, F54, F65, H350,	
\skip .....	b28, b53, b92, b208, b250,	H410, H487, O452, O520, V646, V705	
b295, d33, J358, O391, V316, V490			
skip commands:			
\skip_zero:N .....	n23,	\c_backslash_str .....	i146, i315
U220, U221, U222, U271, U272, U273		\c_hash_str .....	g1293
\skipdef .....	b45, b53, b92, d221	\str_case:nn .....	g1010, i411
\skipzero .....	d221	\str_case:nnTF .....	g2490, i39
\slash .....	b441, f785, f806	\str_case_e:nnTF .....	g2162
\sldefault .....	v691, z24, A94	\str_count:n .....	i158
\slloppy .....	J300, J319, R57, R62	\str_gset:Nn .....	h1437
sloppypar (environment) .....	R62	\str_head:n .....	g2107
\slloppypar .....	R62	\str_if_eq:nn .....	223
\slshape .....	r440,	\str_if_eq:NNTF .....	T518
r800, v689, v690, z22, z23, C22, D627		\str_if_eq:nNTF .....	
\small .....	179	153, 156, e136, g357, g1050, g1055,	
\smallbreak .....	b449, f786, f807	g1374, g1746, g1759, g1761, g1791,	
\smallint .....	A357	g1811, g1813, g1832, g1859, g1887,	
\smallskip .....	b450, o400	g1889, g1914, g1916, g1939, g2226,	
\smallskipamount .....	b449, o400, o403	h184, h276, h316, h348, h350, h425,	
\smash A473, A547, A548, A551, A552, H126		h480, h527, h539, h607, h614, h917,	
\smile .....	A450	h1000, h1080, h1169, h1304, h1336,	
\sourceLaTeXdate .....	c161, S65, S103	i209, i287, i373, i376, T131, T512, T522	
\sp .....	H199		
\space .....	b414	\str_if_eq_p:nn .....	
\spacefactor .....	b439, b440, o129,	.. g416, g417, g418, g419, h543, h800	
o138, o157, o171, o183, o197, o211,			
o416, o429, o434, r70, r73, O531, O533		\str_map_function:NN .....	e140
\spaceskip .....	D6, D612	\str_new:N .....	g19
\spadesuit .....	A334	\str_set:Nn .....	
\span .....	K383	.. g176, g177, g219, g2040, T366, T367	
\special ... 198, 860, 869, 871, 872, 884		\str_tail:n .. 152, g1746, g2104, g2130	
\SplitArgument .....	g2626	\str_uppercase:n .....	g2426
\splitfirstmark .....	V2245		
\SplitList .....	174, g2626	str internal commands:	
\splitmaxdepth .....		\_str_if_eq:nn .....	200, h23
... b377, O469, O488, O506, V2239		\strcmp .....	S283, S299
\splittopskip ... b389, O468, O487, O505		\stretch .....	o456
\sqcap .....	A376	\string .....	845, 848
\sqcup .....	A377	\strut 199, b457, f787, f808, H178, H179, K29	
\sqrt .....	H342	\strutbox .....	b457, w189,
\sqrtsign .....	A529, H71, H342	J384, J401, J418, K186, K187,	
\sqsubset .....	z729	O469, O477, O488, O495, O506, O513	
\sqsubseteqq .....	A399	\subparagraphmark .....	N143
\sqsupset .....	z730	\subsectionmark .....	N143
\sqsupseteq .....	A400	\subset .....	A426
\SS .....	r304, r534, r1149, X571	\subsetneq .....	A428
\ss ... r254, r406, r557, r784, r1124, X571		\subsubsectionmark .....	N143
\sscdefault .....	v536, v609, v706	\succ .....	A420
\sscsshape ... v536, v608, v704, v705, C31		\succceq .....	A423
\stackrel .....	H340	\sum .....	A350
\star .....	A397	\sup .....	H24
		\suppressfloats .....	V2006
		\supset .....	A425

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltruatex.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=ltspacedtx, p=ltlogos.dtx, q=ltfiles.dtx,  
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
X=ltfinal.dtx

\supseteqq ..... A427  
 \surd ..... A336  
 \swarrow ..... A408  
 \swdefault ..... v531, v607, v703  
 \swshape ..... v531, v606, v701, v702, C30  
 \symbol ..... r162, z619  
 \symletters ..... D8, D14, D614, D1139  
 \symoperators ..... A643  
 sys commands:  
     \sys\_if\_engine\_luatex:TF ..... U26

**T**

\T ..... l23, r335, r337, r339, r341, r343, r345, r347, r349, r351, r374, S1446, S1450, S1451  
 \t ..... r282, r741, r849, D157, D158, D185, D189, D191, D203, D206, D208, D688, D859, D1126, D1128  
 tabbing (environment) ..... K71  
 \tabbing ..... K71  
 \tabbingsep ..... K130, K132, K166  
 \tabcolsep ..... K259, K338  
 \tableofcontents ..... 612  
 \tabskip ..... b468, H195, H196, H356, H359, H362, H364, H492, H505, H508, H510, K167, K192  
 tabular (environment) ..... K174  
 \tabular ..... K174  
 \tabular\* ..... K175  
 \tabularnewline ..... K194, K207  
 \tan ..... H15  
 \tanh ..... H17  
 \tau ..... A285  
 \tencirc ..... B10, L125, L678  
 \tencircw ..... B10, L128  
 \tenln ..... B9, L124, L126, L677, L679  
 \tenlnw ..... B9, L127, L129  
 \TeX ..... p1, p12  
 TEX and L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub>  commands:  
     \...-h@k ..... 813  
     \...@without@substitution ..... 454  
     \@ ..... a65, d20, d862, f728, f729, i292, l19, o425, p2, S43, S58, S71, S80, S118, X503  
     \@...hook ..... 205, 206  
     \@@ ..... a331, a332, k15, k19, k20, k21, k22, k24, k27, k28, k30, k31, q656, q672, w510, w512, w513, K238, K239, K240, K250, V10, V11  
     \@@defaultsups ..... u585  
     \@@enc@update ..... r183, u259, u263  
     \@@end 342, a69, a222, f23, q633, q634, G32, G99, G165, W18, X590, X611  
     \@@endpbox ..... K193, K236, K386

\@eqnacr H368, H390, H400, H405, H511  
 \@fileswith@ptions S540, S559, S1003  
 \@hyph ..... f24, f745  
 \@hyphenation ..... r205  
 \@if@newlist V599, V644, V657, V703  
 \@ifdefinable ..... f132, r17  
 \@input ..... a68, f22, q531, q544, q584, G26, G79, S1593, T172, T188, T196, X317  
 \@italiccorr ..... f25, C113, C117  
 \@line ..... J489  
 \@math@bgroup ..... C131, C138  
 \@math@egroup ..... C128  
 \@par ..... 299, 302, f21, m4, n98, n135, G165, G421, G426, G429, G443, G447, G450, J266, J288, J309, K199, N67, N118, V257, I82, I85  
 \@patterns ..... r205  
 \@protect ..... f254, f260, f269  
 \@startpbox ..... K193, K236, K386  
 \@sverb ..... 626, G500  
 \@underline ..... J445, J448, J449  
 \@unprocessedoptions ..... S915, S980, S1057  
 \@warning ..... l215  
 \@Alph ..... 99, s140, s156  
 \DeclareEncodingSubset ..... . D45, D47, D48, D49, D50, D53, D60  
 \DeclareMathDelimiter ... y875, y894  
 \DeclareMathSizes .. u206, u207, u209  
 \Eshack .... o190, O201, O223, O241  
 \IncludeInRelease ..... c68  
 \IncludeInRelease ..... c68  
 \OM ..... b21, b442, b443, f37, f39, o11, o12, o13, o14, o15, o16, o17, o18, o119, u654, u661, w439, w452, H378, K56, N67, N100, N118, N130, N209, N232, V176, V195, V198, V258, I194  
 \OMM 269, b21, O469, O488, O506, V299  
 \OMi ..... j3, V136  
 \OMii ... j3, O53, O122, O194, O216, O241, O311, V295, V1159, V1326  
 \OMiii ..... j3, O55, O124, O313, V298  
 \OMiv j3, O195, O201, O217, O223, V272  
 \Roman ..... s138, s144  
 \TeXversion ..... 2, a326, l28  
 \abspage@last 878, 880, U105, U111, U349, U351, U353, U361, U398, U399  
 \accii ..... z775, J290, J311  
 \acciii ..... z775, J290, J311  
 \acciii ..... z775, J290, J311

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

\@acol ..... K168, K178, K260, K261, K273, K274, K277, K294, K309, K317, K327  
 \@acolampacol ..... K258, K275, K277, K284, K292, K326, K329  
 \@activechar@info ..... V575  
 \@addamp ..... K251, K260, K261, K276, K290, K327, K328  
 \@addfield ..... K43, K53, K86, K93, K125, K140, K142  
 \@addmarginpar ..... V331, V1811  
 \@addtobot ..... V975, V1062, V1129, V1181, V1290, V1349  
 \@addtocurcol ... V328, V1066, V1999  
 \@addtoblcol ..... V854, V1562  
 \@addtofilelist ..... a101, a103, q64, q132, q187, q531, q653, z749, z752, z759, z762, z769, z772, T169, T188, T196, X265, X268, X631  
 \@addtonextcol .. V853, V1386, V2000  
 \@addtopreamble ..... K311, K324, K330, K331, K332, K334, K346  
 \@addtoreset s16, s39, s44, s89, s115, s118  
 \@addtoporbot ..... V1012, V1175, V1343, V1435, V1524  
 \@afterheading ..... N92, N125  
 \@afterindentfalse ..... N45  
 \@afterindenttrue ..... N43, N124, N208, N231  
 \@alph ..... 99, s139, s152, O399  
 \@ampacol ... K258, K275, K286, K329  
 \@arabic s43, s111, s123, s136, s142, O397  
 \@arrayarraycr ..... K203, K204  
 \@argdef ..... f80  
 \@argsrsbox ..... J469  
 \@argtabularcr ..... K210, K211  
 \@array ..... K181, K182  
 \@arrayacol ..... K168, K258  
 \@arrayclassiv ..... K169, K331  
 \@arrayclassz ..... K168, K275  
 \@arraycr ..... K170, K201, K203  
 \@arrayparboxrestore J280, J322, K384  
 \@arrayrule ..... K309, K311, K315, K317, K319, K346  
 \@arstrut ..... K192, K237, K343  
 \@arstrutbox K185, K218, K343, K385  
 \@author ..... N8, N32  
 \@auxout ..... q216, q222, q266, q284, q308, q337, q363, q378, F33, N181, Q7, Q8, Q19, Q29, Q37, Q47, Q64, U360  
 \@backslashchar ..... f219, f469, f592, f594, f598, f599, f604, l234, l236, A266, S1182, S1315, S1404  
 \@badcrerr ..... l277  
 \@badend ..... l247, G297  
 \@badlinearg ..... l267, L165, L177, L188, L189, L193, L242, L247, L257, L262, L275  
 \@badmath . l251, H262, H264, H269, H272, H281, H293, H298, H307, H320, H325, H434, H446, H462, H471  
 \@badpoptabs ..... l255, K85, K151  
 \@badrequireerror ..... S434, S1065  
 \@badtab ..... l258, K22, K87, K108, K114, K121, K148  
 \@begin@tempboxa ..... J27, J42, J203, J266, J470, J478  
 \@begindocumenthook ..... 193, 205, 206, q61, q126, q129, q181, q184, Q33, S1014, S1035  
 \@begindvi .... 865, V623, V682, V710  
 \@begindvibox ..... 860, U438, U439, V86, V711  
 \@beginparpenalty ..... 646, o14, H437, H449, H475, I23, I170  
 \@begintheorem ..... M30, M35  
 \@bezier ..... L683, L684  
 \@bibitem ..... Q3, Q8  
 \@biblabel ..... Q4, Q77  
 \@bitor ..... V15, V881, V901, V937, V960, V1027, V1111, V1121, V1269, V1280, V1422, V1509, V1627, V1752  
 \@botlist ..... V65, V384, V386, V431, V433, V717, V738, V747, V748, V989, V992, V1027, V1121, V1280, V1955, V1983  
 \@botnum ..... O274, V109, V986, V987, V992, V996, V1458, V1463, V1551, V1558, V1947, V1975, V2017  
 \@botroom ..... O275, V110, V989, V992, V1948, V1976  
 \@boxfpsbit .... V2065, V2067, V2072  
 \@break@tfor ... k31, q504, q521, C98  
 \@bsphack ... o36, o125, o340, o356, o374, o390, F32, O52, O121, O310, P6, P18, P23, P35, Q43, Q60, V1882  
 \@caption ..... O12, O14  
 \@captype ..... O5, O9, O12, O40, O88, O109, O157, V2029  
 \@car ..... f53, p14, r89, r110  
 \@carcube ..... f55, f135, f585  
 \@cclv ..... 864, b16, V300, V304, V382, V383, V412, V429, V430, V459, V483, V487, V488, X53

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltruatex.dtx, e=lxexpl.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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssrc.dtx, x=ltfscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

\@cclvi ..... b<sub>21</sub>, b<sub>57</sub>, b<sub>82</sub>, b<sub>93</sub>, b<sub>95</sub>, b<sub>99</sub>, b<sub>159</sub>, b<sub>173</sub>, d<sub>30</sub>, d<sub>58</sub>, S<sub>1180</sub>, S<sub>1313</sub>, S<sub>1402</sub>  
 \@cdr ..... f<sub>53</sub>, f<sub>671</sub>, f<sub>672</sub>, q<sub>250</sub>  
 \@centercr ..... G<sub>320</sub>, G<sub>358</sub>, G<sub>363</sub>, G<sub>368</sub>, G<sub>378</sub>, G<sub>382</sub>, G<sub>386</sub>  
 \@centering ..... H<sub>347</sub>, H<sub>348</sub>, H<sub>356</sub>, H<sub>359</sub>, H<sub>362</sub>, H<sub>504</sub>, H<sub>508</sub>  
 \@cflb ..... V<sub>714</sub>  
 \@cflt ..... V<sub>714</sub>  
 \@changed@cmd ..... r<sub>3</sub>, r<sub>63</sub>, r<sub>223</sub>, u<sub>131</sub>, u<sub>267</sub>, X<sub>462</sub>  
 \@changed@x ..... r<sub>3</sub>, r<sub>211</sub>, r<sub>219</sub>  
 \@changed@x@mouth ..... r<sub>211</sub>, r<sub>219</sub>  
 \@charlb ..... q<sub>384</sub>, q<sub>392</sub>  
 \@charrb ..... q<sub>386</sub>, q<sub>392</sub>  
 \@chclass K<sub>271</sub>, K<sub>272</sub>, K<sub>335</sub>, K<sub>348</sub>, K<sub>353</sub>  
 \@check@IncludeInRelease ..... c<sub>68</sub>  
 \@check@c ..... f<sub>189</sub>, f<sub>191</sub>  
 \@check@eq ..... f<sub>195</sub>, f<sub>196</sub>, f<sub>200</sub>  
 \@checkend ..... G<sub>16</sub>, G<sub>71</sub>, G<sub>213</sub>, G<sub>220</sub>, G<sub>272</sub>, G<sub>287</sub>, G<sub>296</sub>  
 \@chnum ..... K<sub>279</sub>, K<sub>298</sub>, K<sub>335</sub>, K<sub>350</sub>, K<sub>351</sub>, K<sub>352</sub>  
 \@circ .. L<sub>622</sub>, L<sub>640</sub>, L<sub>649</sub>, L<sub>654</sub>, L<sub>657</sub>  
 \@circle ..... L<sub>605</sub>, L<sub>606</sub>  
 \@circlefnt L<sub>125</sub>, L<sub>128</sub>, L<sub>447</sub>, L<sub>491</sub>, L<sub>520</sub>, L<sub>545</sub>, L<sub>615</sub>, L<sub>631</sub>, L<sub>663</sub>, L<sub>678</sub>  
 \@cite ..... Q<sub>16</sub>, Q<sub>75</sub>  
 \@cite@ofmt ..... Q<sub>24</sub>, Q<sub>76</sub>  
 \@citea ..... Q<sub>15</sub>, Q<sub>17</sub>  
 \@citeb ..... Q<sub>18</sub>, Q<sub>19</sub>, Q<sub>20</sub>, Q<sub>23</sub>, Q<sub>24</sub>, Q<sub>46</sub>, Q<sub>47</sub>, Q<sub>48</sub>, Q<sub>49</sub>, Q<sub>63</sub>, Q<sub>64</sub>, Q<sub>65</sub>, Q<sub>66</sub>  
 \@citex ..... Q<sub>13</sub>, Q<sub>14</sub>  
 \@classi ..... K<sub>271</sub>, K<sub>307</sub>  
 \@classii ..... K<sub>271</sub>, K<sub>321</sub>  
 \@classiii ..... K<sub>271</sub>, K<sub>326</sub>  
 \@classiv ..... K<sub>169</sub>, K<sub>180</sub>, K<sub>272</sub>  
 \@classoptionslist ..... S<sub>9</sub>, S<sub>481</sub>, S<sub>496</sub>, S<sub>497</sub>, S<sub>514</sub>, S<sub>726</sub>, S<sub>727</sub>, S<sub>755</sub>, S<sub>756</sub>, S<sub>782</sub>, S<sub>783</sub>, S<sub>1674</sub>  
 \@classv ..... K<sub>272</sub>, K<sub>332</sub>  
 \@classz ..... K<sub>168</sub>, K<sub>179</sub>, K<sub>271</sub>  
 \@cline ..... K<sub>367</sub>  
 \@clnht ..... L<sub>195</sub>, L<sub>196</sub>, L<sub>204</sub>, L<sub>206</sub>, L<sub>208</sub>, L<sub>218</sub>, L<sub>225</sub>, L<sub>273</sub>, L<sub>672</sub>  
 \@clnwrd ..... L<sub>197</sub>, L<sub>203</sub>, L<sub>207</sub>, L<sub>209</sub>, L<sub>210</sub>, L<sub>672</sub>  
 \@cls@pkg ..... h<sub>1356</sub>, S<sub>290</sub>, S<sub>291</sub>, S<sub>304</sub>, S<sub>305</sub>, S<sub>825</sub>, S<sub>871</sub>, S<sub>912</sub>, S<sub>959</sub>, S<sub>989</sub>, S<sub>1041</sub>, S<sub>1050</sub>, S<sub>1052</sub>, S<sub>1069</sub>, S<sub>1527</sub>, S<sub>1602</sub>, S<sub>1624</sub>, S<sub>1654</sub>  
 \@clexception ..... 813, 845, S<sub>31</sub>, S<sub>156</sub>, S<sub>164</sub>, S<sub>220</sub>, S<sub>320</sub>, S<sub>336</sub>, S<sub>348</sub>, S<sub>430</sub>, S<sub>452</sub>, S<sub>463</sub>, S<sub>481</sub>, S<sub>495</sub>, S<sub>513</sub>, S<sub>612</sub>, S<sub>627</sub>, S<sub>651</sub>, S<sub>725</sub>, S<sub>754</sub>, S<sub>781</sub>, S<sub>875</sub>, S<sub>905</sub>, S<sub>932</sub>, S<sub>993</sub>, S<sub>1006</sub>, S<sub>1042</sub>  
 \@clubpenalty ..... q<sub>7</sub>, q<sub>25</sub>, q<sub>94</sub>, q<sub>151</sub>, N<sub>106</sub>, N<sub>135</sub>, I<sub>128</sub>, I<sub>196</sub>  
 \@colht ..... q<sub>22</sub>, q<sub>91</sub>, q<sub>148</sub>, O<sub>273</sub>, O<sub>275</sub>, O<sub>278</sub>, O<sub>284</sub>, O<sub>285</sub>, O<sub>298</sub>, O<sub>299</sub>, V<sub>114</sub>, V<sub>231</sub>, V<sub>242</sub>, V<sub>251</sub>, V<sub>252</sub>, V<sub>387</sub>, V<sub>399</sub>, V<sub>434</sub>, V<sub>447</sub>, V<sub>474</sub>, V<sub>505</sub>, V<sub>535</sub>, V<sub>541</sub>, V<sub>545</sub>, V<sub>555</sub>, V<sub>560</sub>, V<sub>645</sub>, V<sub>704</sub>, V<sub>777</sub>, V<sub>815</sub>, V<sub>859</sub>, V<sub>884</sub>, V<sub>903</sub>, V<sub>943</sub>, V<sub>965</sub>, V<sub>1642</sub>, V<sub>1768</sub>, V<sub>2130</sub>, X<sub>141</sub>  
 \@colnum ..... O<sub>276</sub>, V<sub>111</sub>, V<sub>995</sub>, V<sub>1040</sub>, V<sub>1109</sub>, V<sub>1110</sub>, V<sub>1138</sub>, V<sub>1146</sub>, V<sub>1267</sub>, V<sub>1268</sub>, V<sub>1300</sub>, V<sub>1312</sub>, V<sub>1420</sub>, V<sub>1421</sub>, V<sub>1458</sub>, V<sub>1463</sub>, V<sub>1507</sub>, V<sub>1508</sub>, V<sub>1550</sub>, V<sub>1557</sub>, V<sub>1943</sub>, V<sub>1971</sub>, V<sub>2010</sub>, V<sub>2185</sub>  
 \@colroom ..... q<sub>23</sub>, q<sub>92</sub>, q<sub>149</sub>, V<sub>115</sub>, V<sub>252</sub>, V<sub>273</sub>, V<sub>274</sub>, V<sub>285</sub>, V<sub>288</sub>, V<sub>387</sub>, V<sub>434</sub>, V<sub>777</sub>, V<sub>994</sub>, V<sub>1039</sub>, V<sub>1105</sub>, V<sub>1108</sub>, V<sub>1137</sub>, V<sub>1262</sub>, V<sub>1266</sub>, V<sub>1299</sub>, V<sub>1416</sub>, V<sub>1419</sub>, V<sub>1502</sub>, V<sub>1506</sub>, V<sub>1944</sub>, V<sub>1972</sub>, V<sub>2140</sub>, V<sub>2145</sub>, V<sub>2190</sub>, X<sub>140</sub>  
 \@combinedblfloats V<sub>750</sub>, V<sub>2264</sub>, V<sub>2303</sub>  
 \@combinefloats ..... V<sub>501</sub>, V<sub>714</sub>  
 \@comdblflelt ..... V<sub>750</sub>  
 \@comflelt ..... V<sub>720</sub>, V<sub>736</sub>, V<sub>750</sub>  
 \@cons ..... b<sub>196</sub>, b<sub>213</sub>, f<sub>52</sub>, s<sub>44</sub>, O<sub>193</sub>, O<sub>215</sub>, O<sub>239</sub>, O<sub>379</sub>, V<sub>237</sub>, V<sub>888</sub>, V<sub>907</sub>, V<sub>923</sub>, V<sub>947</sub>, V<sub>949</sub>, V<sub>969</sub>, V<sub>971</sub>, V<sub>1141</sub>, V<sub>1209</sub>, V<sub>1305</sub>, V<sub>1378</sub>, V<sub>1451</sub>, V<sub>1541</sub>, V<sub>1644</sub>, V<sub>1667</sub>, V<sub>1770</sub>, V<sub>1795</sub>, V<sub>1812</sub>, V<sub>1813</sub>, V<sub>2191</sub>  
 \@contentsline@destination ..... 747, N<sub>188</sub>, N<sub>190</sub>, N<sub>197</sub>  
 \@contfield ..... K<sub>50</sub>, K<sub>141</sub>, K<sub>153</sub>  
 \@copy@... ..... 92  
 \@copy@DeclareRobustCommand ..... f<sub>483</sub>, f<sub>520</sub>, f<sub>541</sub>, f<sub>616</sub>, f<sub>619</sub>  
 \@copy@newcommand ..... 86, 93, f<sub>306</sub>, f<sub>484</sub>, f<sub>520</sub>, f<sub>562</sub>, f<sub>591</sub>, f<sub>616</sub>, f<sub>622</sub>  
 \@ctrerr .. l<sub>243</sub>, s<sub>155</sub>, s<sub>159</sub>, s<sub>173</sub>, s<sub>181</sub>  
 \@curfield ..... K<sub>16</sub>, K<sub>41</sub>, K<sub>47</sub>, K<sub>51</sub>, K<sub>52</sub>, K<sub>54</sub>, K<sub>130</sub>, K<sub>131</sub>  
 \@curline ..... K<sub>16</sub>, K<sub>27</sub>, K<sub>39</sub>, K<sub>44</sub>, K<sub>53</sub>, K<sub>54</sub>, K<sub>55</sub>, K<sub>90</sub>, K<sub>91</sub>, K<sub>103</sub>, K<sub>128</sub>, K<sub>129</sub>  
 \@curr@enc ..... r<sub>154</sub>, r<sub>156</sub>

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltruatex.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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfsstrc.dtx, x=ltfscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

\@curr@file 811, 812, 848, 850, q226, q227, q236, q238, q262, q270, q397, q416, q551, q566, S856, S1127, S1132, S1138, S1144, S1148, S1159, S1168, S1195, S1258, S1263, S1269, S1292, S1301, S1327, T267, T341, T343  
 \@curr@file@reqd ..... 848, 850, T267, T343, T347  
 \@currbox ..... b275, b276, b277, O60, O91, O95, O129, O160, O164, O193, O214, O215, O239, O257, O259, O261, O319, O322, O327, O331, V213, V214, V225, V226, V228, V229, V237, V311, V312, V853, V854, V1102, V1104, V1112, V1135, V1139, V1141, V1156, V1197, V1209, V1257, V1260, V1297, V1302, V1305, V1322, V1367, V1378, V1410, V1426, V1440, V1451, V1493, V1530, V1541, V1581, V1585, V1596, V1602, V1604, V1608, V1613, V1622, V1631, V1637, V1644, V1667, V1702, V1706, V1718, V1725, V1727, V1731, V1737, V1747, V1762, V1770, V1795, V1813, V1822, V2035, V2036, V2065, V2095, V2100, V2146, V2149, V2161, V2169, V2186, V2191  
 \@currdir ..... . . . . . 9, a108, a130, a132, a138, a140, a146, a148, a153, a155, a165, a178, a243, a256, a269, S1110, S1132, S1159, S1241, S1263, S1292, S1375  
 \@current@cmd ..... r25, u271  
 \@currentcounter ..... F39, F40, F42, F53, H352, H489, J379, O471  
 \@currentlabel ..... 639, F34, F43, F55, F66, F74, H351, H488, J380, J397, J414, O472, O490, O508  
 \@currenvir ..... . . . . . 283, 294, l249, G3, G176, G231, G248, G281, G297, J149, S1170, S1182, S1190, S1194, S1201, S1303, S1315, S1323, S1327, S1333, S1392, S1404, S1412, S1416, S1422, I112  
 \@currenline ..... l249, G177, G232, G249, G282, G298, J150  
 \@currext .. 812, S30, S42, S57, S70, S79, S117, S316, S319, S320, S335, S336, S347, S348, S452, S463, S474, S481, S495, S513, S542, S622, S635, S637, S645, S659, S821, S822, S823, S828, S834, S841, S846, S848, S853, S859, S867, S873, S875, S882, S885, S890, S893, S896, S898, S899, S901, S905, S914, S916, S917, S922, S925, S928, S932, S938, S951, S956, S957, S962, S968, S972, S974, S975, S977, S979, S981, S982, S985, S991, S993, S1006, S1019, S1042, S1058, S1059  
 \@currlist O193, O215, O379, V67, V311, V388, V391, V435, V438, V1812  
 \@currname ..... 209–211, 245, 810–812, c68, c113, c121, h206, h212, h267, q665, S29, S41, S56, S69, S78, S116, S288, S290, S302, S304, S316, S319, S335, S347, S474, S542, S635, S637, S645, S659, S819, S822, S823, S825, S826, S828, S834, S841, S843, S846, S848, S853, S858, S867, S871, S873, S882, S884, S890, S893, S896, S898, S899, S903, S907, S914, S916, S917, S922, S925, S929, S933, S938, S950, S974, S975, S977, S979, S981, S982, S1019, S1050, S1052, S1059, S1069, S1602, S1624, S1654  
 \@currnamestack ..... 211, 245, 791, 793, 855, h264, S33, S136, T509  
 \@curroptions ..... S474, S482, S532, S551, S1059, S1060  
 \@currpath . 791, 797, 810, 811, 813, S15, S46, S139, S288, S290, S820, S841, S848, S857, S882, S883, S910  
 \@currpkg@reqd ..... 812, S318, S851, S853, S862, S878, S892, S895, S910, S912  
 \@currsize ..... z639  
 \@currtype V119, V878, V879, V880, V881, V898, V899, V900, V901, V1027, V1111, V1121, V1269, V1280, V1422, V1509, V1627, V1752, V2035, V2037, V2038, V2041  
 \@curtab ..... K11, K26, K86, K87, K88, K94, K95, K98, K102, K103, K107, K146, K147  
 \@curtabmar ..... K11, K25, K26, K38, K44, K89, K102, K106, K107  
 \@cd@r ..... a161, a162  
 \@dashbox ..... L324, L325, L326, L327, L328, L331, L337, L339, L349, L351, L352, L353, L354, L358, L362, L365, L382, L384, L385, L386, L387, L390, L394, L396, L405, L407, L408, L409, L410, L413, L417, L422, L674  
 \@dashcnt ..... L317, L319, L320, L321, L322, L323, L336,

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

L338, L341, L343, L344, L345,  
 L347, L348, L361, L364, L376,  
 L377, L378, L379, L380, L381,  
 L393, L395, L398, L399, L400,  
 L401, L403, L404, L416, L421, L674  
 \@dashdim ..... L316,  
 L317, L318, L319, L320, L322,  
 L325, L327, L328, L329, L336,  
 L338, L340, L341, L342, L343,  
 L344, L347, L351, L353, L354,  
 L355, L363, L366, L375, L376,  
 L377, L378, L380, L384, L386,  
 L387, L388, L393, L395, L397,  
 L398, L399, L400, L403, L407,  
 L409, L410, L411, L419, L424, L674  
 \@date ..... N9, N33  
 \@dbflt ..... O32, O264  
 \@dblarg ..... f693, N54, N142, O12  
 \@dbldeferalist ..... O239, V70,  
     V445, V450, V452, V816, V823,  
     V824, V1752, V1795, V1959, V1988  
 \@dblfloat ..... O31  
 \@dblfloatplacement .....  
     ... q31, q100, q158, O280, V401,  
     V449, V1940, V1968, V2269, V2309  
 \@dblflset ..... O26  
 \@dblfpbot ..... O290, O304, V2353  
 \@dblfpsep ..... O289, O303, V2353  
 \@dblftop ..... O288, O302, V2353  
 \@dbltoplist ..... V69, V232, V235,  
     V237, V397, V398, V445, V446,  
     V755, V759, V761, V762, V1639,  
     V1644, V1764, V1770, V1958, V1986  
 \@dbltopnum ..... O283, O297, V107,  
     V127, V238, V240, V766, V1578,  
     V1579, V1643, V1646, V1674,  
     V1679, V1699, V1700, V1769,  
     V1773, V1802, V1807, V1951, V1979  
 \@dbltoproom ..... O284,  
     O286, O298, O300, V108, V1581,  
     V1584, V1585, V1594, V1595,  
     V1598, V1601, V1604, V1608,  
     V1612, V1616, V1621, V1641,  
     V1702, V1705, V1706, V1715,  
     V1716, V1717, V1720, V1724,  
     V1727, V1731, V1736, V1740,  
     V1745, V1746, V1767, V1952, V1980  
 \@dec@text@cmd ..... r3  
 \@declarecommandcopylisthook ..... 91,  
     92, 133, 137, f480, f482, f494, g1018  
 \@declaredoptions ..... S8,  
     S437, S478, S516, S537, S556, S1012  
 \@declareoption ..... S435, S436, S444  
  
 \@defaultfamilyhook .....  
     ... z429, z664, z681, z696, z701, z716  
 \@defaultsubs ..... u539, u573, u585, G46, G86  
 \@defaultunits ..... u214, u218, u219,  
     u220, u235, u328, w179, w181, L13  
 \@defaultunitsset ..... J53, J64,  
     L8, L29, L30, L32, L34, L60, L63,  
     L84, L85, L107, L108, L164, L241,  
     L316, L318, L332, L340, L342,  
     L357, L480, L481, L612, L648,  
     L690, L691, L693, L694, L697,  
     L698, L700, L701, L712, L713,  
     L715, L716, L718, L719, L721, L722  
 \@defdefault@ds ..... S435, S440, S445  
 \@deferlist .....  
     ... V68, V384, V393, V394, V397,  
     V402, V404, V410, V431, V440,  
     V442, V778, V786, V787, V798,  
     V803, V804, V1111, V1209, V1269,  
     V1378, V1422, V1451, V1509,  
     V1541, V1627, V1667, V1957, V1985  
 \@definecounter .....  
     ... s12, s36, H336, M8, M16,  
     O396, O398, I227, I228, I229, I230  
 \@depth ... f26, w191, A558, A559,  
     A561, A562, J444, J488, K187,  
     K219, L227, L300, L303, L324,  
     L333, L383, L391, L727, L783, V1851  
 \@dir .... a160, a163, a165, a167, a168  
 \@disable@packageload@do .. S846, T444  
 \@dischyp ..... f730, f764, J289, J310  
 \@docclearpage ..... V296, V371  
 \@documentclasshook .....  
     ... S3, S731, S759, S786  
 \@doendpe G214, G221, G273, G288, I123  
 \@dofilelist ... q662, q678, G38, G81  
 \@donoparitem ..... I144, I158  
 \@dot ..... L605, L643  
 \@dotsep ..... N215, N238  
 \@dottedtocline ... N200, N226, N227  
 \@downline ..... L297, L301, L306  
 \@downvector ..... L268, L306  
 \@eha ... c180, f280, f319, f347, l219,  
     l237, l239, l241, l250, l252, l282,  
     q223, q267, q285, r52, r84, u28, u58,  
     u102, u144, u187, u253, u339, w106,  
     y25, y70, y99, y172, y240, y270,  
     y302, y416, y437, y469, y510, y555,  
     y560, y615, y733, y737, y741, y776,  
     y780, y784, y841, y851, y936, y941,  
     y944, y976, y979, y1052, y1055,  
     y1058, y1125, y1131, C146, D24,

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

D81, D915, D924, G175, G230,  
 G247, G280, Q51, Q68, V1876, V1892  
 \@ehb ..... l219,  
 l244, l270, l272, l274, V234, V390, V437  
 \@ehc ..... f128, f155, f469, l219,  
 l277, l280, l286, l288, G509, G524,  
 G539, G552, H409, N31, S1654, I220  
 \@ehd ..... c99, c156,  
 e88, l219, l246, l254, l257, l259,  
 l265, y118, K100, K109, O6, S692, S912  
 \@elt ..... 514, 515, 526, f52, q385,  
 r1564, r1566, s20, s35, s53, s56,  
 z141, z152, z154, z155, z180, z205,  
 z207, z508, z518, z742, A17, A25,  
 V8, V11, V15, V27, V30, V31, V32,  
 V33, V38, V39, V40, V41, V42,  
 V43, V44, V45, V47, V51, V57,  
 V58, V59, V60, V498, V720, V731,  
 V736, V746, V758, V760, V788,  
 V805, V825, V844, V857, V864,  
 V915, V918, V927, V1918, V1930  
 \@empty ..... 68, 84,  
 402, 441, 511, 518, 535, 542, 793, k14  
 \@emptycol ..... V198, V245, V248, V277, V281  
 \@end@check@IncludeInRelease ..... c140, c142  
 \@end@tempboxa ..... J36, J45, J208, J279, J476, J486  
 \@enddocument@kernel@warnings ..... G39, G41, G101  
 \@enddocumenthook . G70, S1014, S1036  
 \@endfloatbox O190, O211, O236, O248  
 \@endparenv ..... 654, I120, I123  
 \@endparpenalty ..... 646, o15, H438, H450, H476, I23, I124  
 \@endpbox ..... K193, K236, K266, K333, K384, K387  
 \@endpfalse .. G181, G235, G252,  
 G285, J152, I129, I131, I135, I136, I138  
 \@endpeltrue ..... I138  
 \@endpetrue ..... I124, I126, I134  
 \@endpreamblehook ..... 328  
 \@endtheorem .. M13, M19, M25, M35  
 \@enlargepage .. V1861, V1866, V1868  
 \@ensuredmath ..... H420, H422  
 \@enumctr ..... I234, I237, I238  
 \@enumdepth .. 657, I226, I232, I233, I234  
 \@enumspacing ..... 657  
 \@eqcnt ..... H344,  
 H406, H411, H491, H506, H507, H509  
 \@eqnrcr H357, H375, H412, H413, H493  
 \@eqnnum H338, H339, H410, H424, H483  
 \@eqnsel ..... H344, H505  
 \@eqnswfalse ..... H374  
 \@eqnswtrue .. H346, H353, H411, H490  
 \@eqpen H344, H378, H380, H391, H401  
 \@err@ I37, I41, I44, I52, I64, I68, I71, I79  
 \@esphack ..... o38,  
 o131, o345, o362, o379, o396, F35,  
 O385, P17, P19, P34, Q54, Q70, V1884  
 \@evenfoot .... R12, R15, V613, V672  
 \@evenhead .... R12, R15, V612, V671  
 \@execute@begin@hook .....  
 ..... 616, G178, G182, G185  
 \@expandtwoargs ..... f217,  
 S222, S287, S480, S516, S570, S579  
 \@expast ..... K239, K267  
 \@expl@@@filehook@clear@replacement@flag@@  
 ..... S386, S409, T288, T309, T482  
 \@expl@@@filehook@drop@extension@N  
 ..... 848, T285, T286, T306, T307, T484  
 \@expl@@@filehook@file@pop@@ .....  
 ..... S864, T157, T488, T498  
 \@expl@@@filehook@file@pop@assign@nnnn  
 ..... 845, T162, T490  
 \@expl@@@filehook@file@push@@ .....  
 ..... S847, T151, T486  
 \@expl@@@filehook@if@file@replaced@TF  
 ..... 848, 850, T282, T303, T480  
 \@expl@@@filehook@if@no@extension@nTF  
 ..... T278, T299, T470, T472, T497  
 \@expl@@@filehook@normalize@file@name@w  
 ..... T284, T305, T478  
 \@expl@@@filehook@resolve@file@subst@Cw  
 ..... S384, S407, T281, T302, T476  
 \@expl@@@filehook@set@curr@file@CnNN  
 ..... 810, 848, 850,  
 S383, S406, S813, T341, T347, T474  
 \@expl@@@hook@curr@name@pop@Cw  
 ..... h1524, S89  
 \@expl@@@initialize@all@C h1524, G186  
 \@expl@@@shipout@add@background@box@Cn  
 ..... U406, U491  
 \@expl@@@shipout@add@background@picture@Cn  
 ..... U406, U495  
 \@expl@@@shipout@add@firstpage@material@CnN  
 ..... U406, U488  
 \@expl@@@shipout@add@foreground@box@Cn  
 ..... U406, U493  
 \@expl@@@shipout@add@foreground@picture@Cn  
 ..... U406, U497  
 \@expl@char@generate@Cnn . e141, f709  
 \@expl@cs@argument@spec@Cn .....  
 ..... e138, e149, f610  
 \@expl@cs@prefix@spec@Cn .....  
 ..... e137, e148, f607

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

\@expl@cs@replacement@spec@ON . . . . .  
     e139, e150, f568, f602, f610  
 \@expl@cs@to@str@ON .. e132, e135,  
     e144, e146, f477, f534, f554, f556,  
     f557, f570, f582, f594, f598, f599, f604  
 \@expl@finalise@setup@@ . . . . .  
     e30, X254, X255  
 \@expl@pop@filename@O .. . . . .  
     e26, S88, S92, S98, S109  
 \@expl@push@filename@O .. 791, e24,  
     S37, S39, S52, S54, S64, S76, S96, S102  
 \@expl@push@filename@aux@O .. . . . .  
     245, e25,  
     h1434, S37, S48, S60, S64, S82, S102  
 \@expl@str@if@eq@nnTF . . . . .  
     e136, e147, f441, f443, S287  
 \@expl@str@map@function@NN .. . . . .  
     99, e140, e151, f706  
 \@expl@sys@load@backend@O .. . . . .  
     e21, e23, q17  
 \@extra@page@added .. . . . . G57, U376  
 \@failedlist .. V842, V865, V881,  
     V888, V901, V907, V923, V937, V960  
 \@fcollmadefalse .. . . . . V833  
 \@fcollmadetrue .. . . . . V921  
 \@file-subst@{file} .. . . . . 846  
 \@filef@und .. . . . .  
     844, q436, q457, q478, q501, q518,  
     q531, q584, T155, T172, T188, T196  
 \@filehook@file@push .. . . . . 811  
 \@filehook@set@CurrentFile .. . . . .  
     q314, S849, T152, T330  
 \@filelist .. . . . .  
     q63, q131, q186, q652, q653, q664,  
     z749, z759, z769, X265, X615, X631  
 \@filesfalse .. . . . .  
     ... q199, S1106, S1107, S1237, S1238  
 \@fileswith@pti@ns .. . . . . S434,  
     S540, S559, S717, S718, S722, S724,  
     S752, S753, S779, S780, S807, S1003  
 \@fileswith@ptions .. . . . .  
     S712, S713, S715, S719  
 \@fileswithoptions .. . . . .  
     S612, S619, S627, S710  
 \@filestrue .. . . . .  
     ... q5, S1089, S1092, S1145,  
     S1149, S1220, S1223, S1278, S1282  
 \@finalstrut .. . . . . J384, J401,  
     J418, J487, K385, O477, O495, O513  
 \@firststampfalse .. K254, K277, K294  
 \@firststamptrue .. . . . . K262  
 \@firstcolfirstmark .. . . . .  
     V2246, V2247, V2251  
 \@firstcoltopmark .. . . . . V2244, V2252

\@firstcolumnfalse .. . . . . V2236, V2281  
 \@firstcolumntrue .. . . . . q28,  
     q97, q155, V98, V207, V2255, V2287  
 \@firstofone .. 91, 178, d12, d100,  
     d108, d166, e34, e61, e76, f212,  
     f465, f470, f471, f474, f475, q125,  
     q180, q404, r68, r153, w346, y53,  
     y81, y146, y221, y251, y282, y886,  
     G69, H418, K372, O10, Q18, Q46,  
     Q63, S838, S865, T321, T336, X345  
 \@firstoftwo .. . . . .  
     338, 340, a87, c82, e35, e43, f212,  
     f442, f444, f466, f537, f587, f645,  
     f655, f665, f692, f716, q458, q502,  
     q519, r133, s191, s196, y890, z510,  
     D69, D836, D887, D903, F19, R16,  
     S161, S193, S205, S228, S246, S1663  
 \@firsttab .. . . . .  
     ... K2, K74, K75, K76, K106, K118  
 \@flcheckspace .. V989, V1025, V2136  
 \@flfail .. . . . .  
     V865, V916, V937, V947, V960, V969  
 \@float .. . . . . O26, O32  
 \@floatboxreset .. O101, O170, O174  
 \@floatpenalty .. . . . .  
     O3, O53, O55, O58, O122, O124,  
     O127, O191, O194, O199, O201,  
     O212, O216, O221, O223, O237,  
     O241, O311, O313, O317, O321, O379  
 \@floatplacement .. . . . .  
     ... q31, q100, q158, O271, V149,  
     V209, V253, V477, V1941, V1969  
 \@fisetnum .. . . . .  
     ... V986, V1022, V1109, V1267,  
     V1420, V1507, V1578, V1699, V2104  
 \@flsettextmin .. . . . .  
     V1085, V1237, V1406, V1489, V2120  
 \@flstop .. . . . . V2006  
 \@flsucceed .. . . . .  
     ... V858, V866, V915, V949, V971  
 \@fltovf .. . . . . l273, O93, O162, O322  
 \@flupdates .. . . . . V992, V1037, V2182  
 \@flushglue .. . . . . j17,  
     G359, G363, G369, G379, G382,  
     G387, G420, G442, J296, J317, I76  
 \@fnssymbol .. . . . . s141, s160  
 \@font@aliasinfo .. . . . . w539  
 \@font@info .. . . . . u133,  
     u171, u177, u366, u383, u621,  
     v478, w30, w38, w46, w74, w87,  
     w154, w200, w214, w225, w239,  
     w255, w261, w274, w281, w288,  
     w293, w303, w315, w327, w491,  
     w503, w508, w515, w545, w558,

**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=ltspacedtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

w566, y180, y197, y206, y312, y327, y374, y420, y489, y495, y539, y552, y635, y724, y767, y832, y926, y1093, y1122, D55, D56, D99, X464  
 \@font@series@contextfalse z502, z538  
 \@font@series@contexttrue ..... z521, z525, z537  
 \@font@shape@subst@warning . 446, v445, v448, v453, v511, v658, v661  
 \@font@warning ..... u3, u535, u540, u567, u574, v456, w19, w33, w41, w49, w61, w77, w476, w490, w502, w507, w514, w557, w565, x30, G43, G83, X286  
 \@fontenc@load@list ..... r1565, z742, A17, A25  
 \@fontswitch ..... C126, C128  
 \@footnotemark ..... O454, O460, O522, O528, O529, O555  
 \@footnotetext ..... J349, O454, O460, O461, O538, O544  
 \@for ..... k16, q232, q304, q359, q664, Q16, Q45, Q62, S230, S247, S478, S497, S514, S532, S537, S551, S556, S592, S602, S1060, S1097, S1228  
 \@forced@seriesfalse v394, v407, w140  
 \@forced@seriestrue ..... v398, v409  
 \@forloop ..... k19, k20  
 \@fornoop ..... k15, k23, k29  
 \@fortmp ..... k17, k18, k26, S590, S592, S1096, S1097, S1227, S1228  
 \@fpbot ..... O290, O304, V863, V2347  
 \@fpmin . O278, O287, O301, V113, V920, V1949, V1977, V2199, V2216  
 \@fps ..... O41, O42, O44, O47, O64, O110, O111, O113, O116, O133, V2027, V2029, V2032  
 \@fpsadddefault ..... O45, O48, O114, O117, V2024  
 \@fpsep ..... O289, O303, V861, V870, V942, V964, V2347  
 \@fpstype ..... V983, V1004, V1005, V1019, V1050, V1051, V1075, V1077, V1080, V1082, V1133, V1189, V1190, V1225, V1228, V1231, V1234, V1295, V1357, V1358, V1396, V1398, V1401, V1403, V1477, V1480, V1483, V1486, V1575, V1590, V1592, V1610, V1619, V1655, V1656, V1696, V1711, V1713, V1733, V1743, V1782, V1783, V2020, V2036, V2038, V2040, V2043, V2044, V2045, V2047, V2048, V2052, V2053, V2055, V2056, V2090, V2092, V2094, V2106, V2108, V2122, V2124, V2154, V2157, V2168  
 \@ftptop .... O288, O302, V860, V2347  
 \@frameb@x ..... J179, J207, J209  
 \@framebox ..... J186, J193, J197  
 \@framepicbox ..... J186, J193, J230  
 \@freelist ..... b196, b213, b275, O60, O129, O319, O320, V29, V34, V48, V56, V213, V499, V732, V747, V761, V866, V1812, V1813  
 \@getcirc ..... L437, L485, L514, L541, L613, L629  
 \@getfpsbit ..... V980, V1016, V1572, V1693, V2063  
 \@getlarrow ..... L266, L274, L276  
 \@getlinechar ..... L190, L229  
 \@getpen ..... o34, o37, o46, o117  
 \@getrarrow ..... L267, L274, L283  
 \@glossaryfile ..... P21, P22, P31  
 \@gnewline ..... o95, o101, o108, o111  
 \@gobble ..... 526, 811, c160, d11, d98, e33, e65, 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, o75, o483, q64, q132, q187, q411, q413, q652, r29, u536, u569, w345, x26, y28, y30, y378, y389, y453, y500, y501, y530, y536, y544, y549, y567, y581, y591, y600, y613, y630, y639, y713, y715, y719, y727, y761, y770, y822, y824, y835, y919, y929, y1010, y1015, y1084, y1115, z139, z180, z518, z752, z762, z772, D864, G239, N143, N144, N145, N146, N147, N182, O7, Q11, Q25, Q26, S680, S844, S1081, S1141, S1168, S1272, S1301, S1385, S1390, S1464, S1608, S1620, T261, V619, V620, V621, V678, V679, V680, V927, V1932, V2200, V2217, X268, X397, X554, X563, X631  
 \@gobble@AddToHook@args h1540, h1541  
 \@gobble@IncludeInRelease ..... c68  
 \@gobble@RemoveFromHook@arg ..... h1543, h1544  
 \@gobblecr ..... o481, o482  
 \@gobblefour ..... f208, y24, y375, y491, y493, y497, y499, y509, y513, y637, y689, S1170, S1303, S1392

**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=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=lpageno.dtx, F=lxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

\@gobblethree 793, f208, f596, f609, S128  
 \@gobbletwo ..... e123, e128, f175, f176, f208, k12, q32, q101, q159, u541, u575, y132, G23, G44, G76, G84, R11, R13, S1140, S1271, S1384, T260, X292  
 \@gttempa ..... f126, f127, f181, f183, f528, f536, q596, q597, q603, q604, q605, q630, q631, q633, q634, q635, K3, K5, K6, K7, K8, S286, S288, S301, S302, S321, S323, S337, S339, S349, S351  
 \@halfwidth ..... L2, L126, L129, L131, L227, L299, L302, L324, L333, L349, L361, L364, L383, L391, L405, L416, L421, L680, L706, L725, L726, L727, L766, L781, L782, L783  
 \@haligno .. K170, K174, K177, K191  
 \@hangfrom ..... N66, N117, N138  
 \@height ..... b436, f26, o350, o358, o384, o392, r293, r295, w190, A340, A558, A559, A561, A562, J163, J168, J216, J226, J444, J488, K186, K219, K359, K376, L227, L300, L303, L324, L333, L351, L359, L383, L391, L407, L414, L590, L600, L726, L782, V1851  
 \@highpenalty ..... o118, X3  
 \@hightab ... K11, K21, K23, K74, K86, K95, K96, K111, K146, K147  
 \@hline ..... L167, L179, L226, L265  
 \@holdpg ..... V122, V300, V302, V303, V308, V309, V310  
 \@hspace ..... o437, o438, o453  
 \@hspacer ..... o437, o452  
 \@hvector ..... L244, L259, L265  
 \@icentercr ..... G337, G338  
 \@iden ..... f215  
 \@if ..... f171, f172, f174  
 \@if@DeclareRobustCommand ..... 93, f483, f507, f518, f519, f523, f614, f615, f618, i125  
 \@if@newcommand ..... 86, 93, 95, 258, f304, f484, f508, f519, f561, f573, f578, i126, i137  
 \@if@pti@ns S222, S225, S227, S244, S245  
 \@if@ptions S219, S220, S221, S823, S957  
 \@ifatmargin ..... K55, K106  
 \@ifbothcounters ..... s61, s72, s80, s88, s104, s106, s115, s117  
 \@ifclasslater .... S163, S171, S178  
 \@ifclassloaded ... S155, S266, S275  
 \@ifclasswith .... S219, S268, S277  
 \@ifdefinable ..... f84, f86, f130, f132, f238, r14, r17, s11, t3, z618, J115, M7, M15, M22  
 \@iffileonpath ..... q432, q453, q474, q491, q512  
 \@ifl@aded ..... 811, S155, S156, S157, S822, S938, S956  
 \@ifl@t@r ..... r1530, S169, S176, S182, S186, S188, S199, S200, S701  
 \@ifl@ter ..... r1569, r1570, S163, S164, S181, S867, S985  
 \@ifl@ter@Q ..... r1569, r1570  
 \@ifnch ..... e49, e50, f677, f679, f691  
 \@ifnextchar ..... a98, e45, f673, f678, f692, o93, o482, q544, s13, w411, G336, H342, J9, J11, J18, J20, J26, J47, J121, J122, J128, J129, J136, J140, J185, J186, J192, J193, J198, J231, J239, J247, J254, J258, J327, J331, J335, J428, J433, J456, J463, J468, K57, K181, K203, K210, L23, L132, L143, L453, M3, M5, M28, O27, O264, O324, O452, O519, O536, Q3, Q13, S293, S307, S696, S711, S716, S1090, S1093, S1221, S1224, V209, V2008, I143  
 \@iforloop ..... k21, k22  
 \@ifpackagelater 788, S163, S170, S177  
 \@ifpackageloaded ..... 788, S155, S265, S274, V1992  
 \@ifpackagewith 788, S219, S267, S276  
 \@iframebox ..... J199, J200, J201  
 \@framepicbox ..... J231, J232  
 \@ifstar ..... f73, f692, o59, o71, o330, o437, s102, s113, u206, x121, G325, G332, G563, G572, H377, K56, K202, K209, L142, L605, N52, N142, S435, S475, V1856  
 \@ifundefin@d@i . f635, f636, f653, f656  
 \@ifundefin@d@ii .... f635, f638, f641  
 \@undefined f127, f134, f154, f161, f183, f194, f277, f283, f316, f322, f344, f350, f380, f397, f478, f630, g2651, s3, s7, s16, s50, s62, s64, u100, u186, w424, y410, D54, D866, F23, G141, G158, G174, G229, G246, G279, M21, Q20, Q48, Q65, R3, R7, S128, S153, S388, S394, S411, S423, S499, S533, S552, T451  
 \@ignorefalse ..... G4, G180, G216, G223, G234, G251, G274, G284, G289, O384  
 \@ignoretrue ..... o200, o213, G4, G7, H335, H338, H371, H514

**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=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

\@iiiminipage ..... J329, J333, J336, J337, J338  
 \@iiiparbox ..... J241, J249, J256, J259, J260, J261, J366  
 \@iiiminipage ..... J332, J334  
 \@iinput ..... 340, q544, q545  
 \@iiparbox ..... J255, J257  
 \@iirsbox ..... J468, J477  
 \@imakebox ..... J26, J41, J138  
 \@imakepicbox ... J47, J48, J143, J233  
 \@iminipage ..... J328, J330  
 \@include .. 332, q227, q270, q286, q290  
 \@includeinreleasefalse ..... c71, c76, c130, c138, f768  
 \@includeinreleasetrue ..... c120  
 \@index ..... P18, P19, P35  
 \@indexfile ..... P4, P5, P14  
 \@inlabel ..... 645  
 \@inlabelfalse ..... V163, V190, I28, I104, I184  
 \@inlabeltrue ..... I28, I178  
 \@inmatherr ..... I283, L605, I112, I142  
 \@inmathwarn ..... r3  
 \@inpenc@test ..... X342, X409  
 \@input ..... 835, q34, q103, q161, q298, q353, q583, N152, X634  
 \@input@ ..... q318, q341, q368, q585, u393, Q31  
 \@input@file@exists@with@hooks T143  
 \@inputcheck ..... 820, a70, a191, a192, a195, a203, b307, e38, e39, e42, f38, f45, q427, q428, q435, q448, q449, q456, q469, q470, q477, q499, q500, q503, q516, q517, q520, S1129, S1130, S1164, S1260, S1261, S1297, S1372, S1373, S1380  
 \@insertfalse ..... V1073, V1223, V1394, V1475, V1570, V1691  
 \@inserttrue ..... V999, V1044, V1161, V1329, V1649, V1776  
 \@invalidchar ..... I288  
 \@iparbox ..... J240, J248, J253  
 \@irsbox ..... J456, J463, J468, J469  
 \@isavebox ..... J136, J137  
 \@isavepicbox ..... J141, J142  
 \@ishortstack ..... L133, L141  
 \@istackcr ..... L143, L144  
 \@itabcr ..... K57, K58  
 \@item ..... I143, I156  
 \@itemdepth . 657, I241, I243, I244, I245  
 \@itemfudge ..... K38, K44, K82  
 \@itemitem ..... I245, I248  
 \@itemlabel ..... I44, I96, I143  
 \@itempenalty ..... 646, o16, I23, I175  
 \@itemspacing ..... 657  
 \@iwhiledim ..... k7  
 \@iwhilenum ..... k3  
 \@iwhilesw ..... k10  
 \@ixpt ..... u717  
 \@ixstackcr ..... L142  
 \@kernel@... ..... 791  
 \@kernel@after@hook ..... 193  
 \@kernel@after@begindocument ..... e7, i60, q58, q75  
 \@kernel@after@begindocument@before ..... 328, q16, q75, v731  
 \@kernel@after@enddocument ..... e1, G15, U350  
 \@kernel@after@enddocument@afterlastpage ..... e1, h293, G19, U356  
 \@kernel@after@para@after ..... 292, n8, n64  
 \@kernel@after@para@end ..... 291, 292, n8, n57  
 \@kernel@after@shipout@background ..... 867, U72, U158  
 \@kernel@after@shipout@lastpage ..... 879, U114, U119, U158, U367, U372  
 \@kernel@before@hook ..... 193  
 \@kernel@before@begindocument ..... e7, q56, q75, U396  
 \@kernel@before@enddocument ..... 612, G13, G105  
 \@kernel@before@para@before ..... 291, 292, n8, n19  
 \@kernel@before@para@begin ..... 291, 292, n8, n27  
 \@kernel@before@shipout@background ..... 867, U68, U70, U158  
 \@kernel@currpathstack ..... 793, S45, S47, S91, S123  
 \@kernel@make@file@csname ..... T280, T283, T301, T304, T330  
 \@kernel@rename@newcommand .. 86, 88, f285, f302, f339, f367, f372, f385  
 \@killglue . L59, L73, L103, L115, L123  
 \@kludgeins ..... V319, V320, V321, V323, V376, V377, V423, V424, V502, V518, V519, V525, V526, V527, V536, V552, V556, V566, V1852, V1883  
 \@labels ..... 645, I27, I146, I147, I189, I206, I207  
 \@largefloatcheck ..... O192, O213, O238, O256  
 \@lastchclass ..... K262, K272, K273, K275, K283, K308, K322, K326, K335, K348, K349

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

```

\@latex@error c97, c156,
c180, 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,
q223, q267, q285, r50, r84, u5, u28,
u58, u102, u144, u187, u253, u339,
w105, x100, x111, y23, y68, y97,
y117, y170, y238, y268, y300, y323,
y339, y416, y437, y469, y509, y513,
y555, y560, y615, y683, y689, y733,
y737, y741, y776, y780, y784, y841,
y851, y936, y941, y944, y976, y979,
y1052, y1055, y1058, y1125, y1131,
z49, z60, z82, z93, z600, z720, C143,
G175, G230, G247, G280, G509,
G524, G539, G551, H409, K100,
K109, N31, O6, O83, Q51, Q68,
S623, S674, S687, S824, S911, S958,
S1049, S1066, S1074, S1079, S1101,
S1154, S1232, S1287, S1623, S1653,
V234, V390, V1874, V1891, I219
\@latex@info 281, f224, f298,
f336, f364, f730, l163, l208, r85, D76
\@latex@info@no@line
..... l163, l209, U89, V576
\@latex@note
..... l190, u33
\@latex@note@no@line l190, S1131, S1137
\@latex@warning
..... l163, l215,
q339, r55, F14, L449, O260, Q22,
Q49, Q66, S1193, S1200, S1326,
S1332, S1415, S1421, U172, V2030
\@latex@warning@no@line
..... e98, f202, l163, l216, q19,
q88, q145, q650, F8, F26, F27, G51,
G58, G91, N32, S289, S303, S702,
S868, S986, S1158, S1262, S1268,
S1291, S1374, S1381, S1448, S1525,
U79, U98, V243, V275, V1827, V2096
\@latexbug
..... l275, V333, V1813
\@latexerr
..... l215, V437
\@latexrelease@catcode@null g6, g2652
\@lbibitem
..... Q3, Q4
\@ldots
..... A504, A506
\@leftcolumn
..... V121, V2237, V2258, V2282, V2291
\@leftmark
..... R16, R50
\@let@token
..... e49, e51,
f677, f680, f683, f691, o410, o411,
o418, C83, C96, H243, H245, H248
\@align
..... H195, H197
\@linechar
..... L190, L191, L192,
L196, L197, L199, L204, L206,
L207, L208, L209, L211, L215,
L216, L219, L220, L225, L272, L670
\@linenft
..... L124, L127,
L190, L265, L273, L304, L307, L677
\@linelen . L164, L165, L176, L177,
L203, L210, L219, L221, L226,
L227, L228, L241, L242, L256,
L257, L300, L303, L305, L306, L671
\@list
..... 646
\@listctr
..... 645, Q9, I202, I225
\@listdepth
..... 645, J350, I23, I35, I38, I43, I99
\@listfiles q62, q130, q185, q656, q671
\@listi
..... 646
\@listii
..... 646
\@listvi
..... 646
\@loadwithoptions
..... S629, S651, S661, S670
\@lowpenalty
..... o117, X3
\@ltab
..... K71, K106
\@m
..... 269, b21, b396, b398,
b399, b432, b433, o288, o429, o434,
q45, q114, q172, L213, L217, Q17, I80
\@mainaux
..... q3,
q37, q38, q106, q107, q164, q165,
q216, q298, q337, q353, q378, G22, G75
\@makebox
..... J11, J20, J25
\@makecaption
..... O24
\@makecol
..... V261, V413, V460, V480
\@makefcolumn
..... V393, V394,
V402, V404, V440, V442, V450,
V452, V2195, V2197, V2213, V2214
\@makefnmark
..... O400, O532
\@makefntext
..... J383, J400, J417, O476, O494, O512
\@makeother
..... a76, a97, a126,
f695, f696, u406, u407, u408, u409,
u410, u411, u412, u413, u414, u415,
u416, G431, G452, G545, G560,
G570, S375, S376, S1175, S1308, S1397
\@makepicbox J10, J19, J46, L366, L424
\@makespecialcolbox
..... V503, V522
\@marbox
..... O320, O322,
O326, O330, O331, O379, V1812,
V1822, V1825, V1833, V1835,
V1836, V1838, V1839, V1840, V1849
\@marginparreset .. O343, O361, O370
\@markright
..... R33, R48
\@maxdepth
..... q60,
q128, q183, V91, V486, V514, X138
\@maxtab
..... K2, K94
\@medpenalty
..... o118, X3
\@meta@family@list
..... z116, z141, z153, z242, z506

```

**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=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,
X=ltfinal.dtx

```

\@midlist . V66, V499, V500, V1027,
 V1029, V1141, V1305, V1956, V1984
\@minipagefalse J323, J325,
 J363, O187, O250, O345, O363, I181
\@minipagerestore J351, J353
\@minipagetrue J324, O186
\@minus f26,
 V2340, V2341, V2342, V2345, V2346
\@missing@onefilewithoptions ...
 S842, S881, S999
\@missingfile@area ...
 q557, q598, q611, S883
\@missingfile@base ...
 q557, q599, q612, S884
\@missingfile@ext ...
 q557, q600, q613, S885
\@missingfileerror 340, 341,
 811, q552, q567, q577, q586, S882, S979
\@mkboth R11, R13
\@mklab I45, I140
\@mkpream K189, K234, K262
\@mparbottom ...
 O387, O388, V118, V476,
 V1823, V1831, V1832, V1833, V1834
\@mpargs J342, J366
\@mparswitchfalse V102
\@mpfn . J348, O452, O457, O541, O545
\@mpfootins J357, J358, J361, J367,
 J374, J375, J392, J393, J409, J410
\@mpfootnotetext J349, J369
\@mplistdepth J350, J367
\@multicnt K370, K372,
 K373, K374, K381, K382, K383,
 L103, L104, L106, L115, L116,
 L118, L667, L704, L706, L707,
 L708, L710, L711, L717, L723,
 L734, L738, L764, L766, L768,
 L770, L771, L775, L779, L790, L794
\@multiplelabels q33, q102,
 q160, F25, F31, G49, G55, G89, G95
\@multiput L86, L95, L98
\@multispan K371, K375, K379
\@namedef ...
 .. f50, q391, u135, u136, u160, v5,
 v525, w418, y360, D57, D93, D869,
 F28, G208, G218, G245, G271,
 G483, G492, H413, H414, K175,
 M12, M13, M18, M19, M23, M24,
 M25, S1092, S1223, T449, X466, X467
\@nameuse f51, q335, q376,
 q390, M23, R5, T452, V607, V665
\@nbitem I168, I221
\@ne 269, b16
\@needsf@rmat S697, S700, S705
\@needsformat S685, S695, S699
\@negargfalse L186
\@negargtrue L185
\@newcommand f79, f80
\@newctr s13, s15, M8
\@newenv f150, f151, f160
\@newenva f148, f149
\@newenvb f150, f151
\@newl@bel F22, G24, G77, Q10
\@newline o94, o96
\@newlistfalse ...
 V600, V658, I29, I33, I108, I182
\@newlisttrue I29, I33, I87
\@next b275, O60, O129, O319, O320,
 V9, V213, V311, V877, V897, V1812
\@nextchar ...
 K269, K270, K330, K331, K332
\@nil a161, a162, c13, c19, c80, c101,
 c102, c114, c115, c162, c163, c164,
 f53, f54, f58, f63, f135, f439, f440,
 f585, f671, f672, k13, k19, k27, p14,
 q246, q247, q248, r89, r110, r996,
 r1000, r1054, r1066, r1068, u358,
 u369, u485, u500, u604, u607, u608,
 u616, v431, v433, v465, v467, v645,
 v647, v671, v673, w350, w351,
 w353, w366, w372, w376, w377,
 w413, w434, w439, w519, w533,
 x26, x44, x53, x57, y40, y479,
 y487, y520, y1136, y1138, C58,
 C62, K367, K368, S90, S91, S97,
 S105, S108, S115, S140, S189, S190,
 S201, S202, S212, S213, S215, S384,
 S407, S450, S455, S571, S591, S595,
 S601, S605, S793, S802, S816, S817,
 S1498, S1503, S1506, S1508, S1509,
 S1524, S1544, S1561, S1615, S1637,
 S1661, T168, T176, T336, T378, T380
\@nmbrlistfalse I33, I46, I91
\@nmbrlisttrue I225
\@nnil f417, f423,
 k13, k20, k21, k22, k28, q247, q248,
 u214, u218, u219, u220, u235, w179,
 w181, w345, w347, w359, w361,
 w366, w380, w382, w389, w400,
 w401, w403, w434, w439, L13,
 S738, S739, S746, S766, S767, S774
\@no@font@optfalse x17, x129
\@no@lnbk o9, o10, o40
\@no@pgbk o7, o8, o32
\@nobreakfalse ...
 .. o120, o122, N94, N129, N157,
 O182, V165, V192, V1150, V1316, I193
\@nobreaktrue o121, N126, O181

```

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

\@nocnterr ..... [I240](#)  
 \@nocounterr ..... [I240](#), [s4](#), [s8](#), [s16](#), [s62](#), [s64](#), [M21](#)  
 \@nodocument ..... [I245](#), [n101](#), [q68](#), [q136](#), [q191](#), [q295](#),  
     [G166](#), [O39](#), [O108](#), [V156](#), [V183](#), [V212](#)  
 \@noitemargfalse ..... [I32](#), [I200](#)  
 \@noitemargtrue ..... [I32](#), [I143](#)  
 \@noitemerr ..... [I278](#),  
     [o248](#), [o266](#), [o303](#), [o326](#), [I69](#), [I81](#), [I107](#)  
 \@noligs [G432](#), [G453](#), [G561](#), [G571](#), [G582](#)  
 \@nolnerr . [I238](#), [o42](#), [o113](#), [G324](#), [G331](#)  
 \@nomath ..... [u1](#),  
     [u337](#), [z551](#), [z586](#), [z592](#), [z613](#), [z615](#), [z637](#)  
 \@noparitemfalse ..... [I30](#), [I145](#)  
 \@noparitemtrue ..... [I30](#), [I66](#)  
 \@noparlistfalse ..... [I31](#), [I70](#)  
 \@noparlisttrue ..... [I31](#), [I67](#)  
 \@normalcr ..... [o52](#), [o92](#), [J322](#)  
 \@normalsize ..... [S4](#), [S5](#)  
 \@noskipsec ..... [645](#)  
 \@noskipsecfalse .....  
     ... [q54](#), [q123](#), [q178](#), [N98](#), [V158](#), [V185](#)  
 \@noskipsectrue ..... [N38](#), [N95](#)  
 \@notdefinable .....  
     ... [f136](#), [f137](#), [f141](#), [f466](#), [I232](#)  
 \@notprerr ..... [I281](#), [q66](#), [q134](#), [q189](#)  
 \@nthm ..... [M3](#), [M4](#)  
 \@nxtabmar ..... [K11](#), [K21](#), [K23](#),  
     [K25](#), [K75](#), [K111](#), [K112](#), [K118](#), [K119](#)  
 \@obsoletefile ..... [q649](#)  
 \@oddfoot .....  
     ... [R11](#), [R14](#), [R15](#), [V124](#), [V610](#), [V669](#)  
 \@oddhead . [R11](#), [R14](#), [V123](#), [V610](#), [V669](#)  
 \@onefilewithoptions .....  
     ... [842](#), [S730](#), [S734](#),  
         [S740](#), [S758](#), [S762](#), [S768](#), [S785](#), [S789](#),  
         [S795](#), [S811](#), [S812](#), [S879](#), [S946](#), [S1474](#)  
 \@onelevel@sanitize . [f697](#), [O42](#), [O111](#)  
 \@onfilewithoptions ..... [814](#)  
 \@onlypreamble .....  
     . [f66](#), [f188](#), [f190](#), [f199](#), [f207](#), [h1450](#),  
         [q196](#), [q205](#), [q242](#), [q651](#), [q677](#), [r23](#),  
         [r24](#), [r61](#), [r62](#), [r66](#), [r125](#), [r145](#), [r189](#),  
         [r190](#), [r204](#), [u17](#), [u115](#), [u117](#), [u123](#),  
         [u139](#), [u167](#), [u182](#), [u203](#), [u208](#), [u250](#),  
         [u512](#), [w419](#), [x28](#), [x36](#), [x42](#), [x79](#),  
         [x83](#), [x88](#), [x93](#), [x98](#), [x108](#), [x126](#),  
         [x127](#), [x128](#), [x134](#), [x138](#), [x142](#), [y17](#),  
         [y19](#), [y44](#), [y46](#), [y107](#), [y116](#), [y136](#),  
         [y353](#), [y354](#), [y362](#), [y402](#), [y440](#), [y442](#),  
         [y444](#), [y457](#), [y472](#), [y519](#), [y521](#), [y563](#),  
         [y602](#), [y618](#), [y695](#), [y789](#), [y798](#), [y854](#),  
         [y857](#), [y860](#), [y880](#), [y893](#), [y947](#), [y982](#),  
     ... [y993](#), [y1007](#), [y1061](#), [y1081](#), [y1085](#),  
         [y1149](#), [C140](#), [C141](#), [D58](#), [D59](#), [D60](#),  
         [D870](#), [F30](#), [P12](#), [P29](#), [Q44](#), [Q61](#),  
         [S10](#), [S13](#), [S85](#), [S112](#), [S120](#), [S122](#),  
         [S154](#), [S294](#), [S357](#), [S363](#), [S428](#), [S431](#),  
         [S432](#), [S443](#), [S444](#), [S445](#), [S470](#), [S476](#),  
         [S489](#), [S526](#), [S544](#), [S564](#), [S584](#), [S608](#),  
         [S613](#), [S617](#), [S620](#), [S628](#), [S649](#), [S652](#),  
         [S662](#), [S681](#), [S694](#), [S699](#), [S705](#), [S714](#),  
         [S719](#), [S807](#), [S879](#), [S1004](#), [S1013](#),  
         [S1021](#), [S1022](#), [S1040](#), [S1047](#), [S1056](#),  
         [S1063](#), [S1064](#), [S1072](#), [S1077](#), [S1082](#),  
         [S1454](#), [S1455](#), [S1456](#), [S1457](#), [S1459](#)  
 \@opargbegintheorem ..... [M32](#), [M35](#)  
 \@opcol ..... [V262](#),  
     [V270](#), [V394](#), [V413](#), [V442](#), [V460](#), [V465](#)  
 \@options ..... [S563](#)  
 \@othm ..... [M3](#), [M20](#)  
 \@outerparskip . [I8](#), [I88](#), [I117](#), [I152](#), [I222](#)  
 \@outputbox ..... [V120](#), [V483](#),  
     [V485](#), [V505](#), [V508](#), [V509](#), [V529](#),  
     [V531](#), [V532](#), [V537](#), [V540](#), [V545](#),  
     [V547](#), [V554](#), [V560](#), [V562](#), [V636](#),  
     [V695](#), [V723](#), [V729](#), [V739](#), [V740](#),  
     [V763](#), [V770](#), [V856](#), [V859](#), [V862](#),  
     [V868](#), [V869](#), [V2237](#), [V2241](#), [V2242](#),  
     [V2256](#), [V2262](#), [V2282](#), [V2288](#), [V2297](#)  
 \@outputdblcol .....  
     . [V468](#), [V2232](#), [V2234](#), [V2278](#), [V2279](#)  
 \@outputpage .. [V403](#), [V452](#), [V470](#),  
     [V590](#), [V2266](#), [V2271](#), [V2304](#), [V2312](#)  
 \@oval ..... [L453](#), [L471](#)  
 \@ovbtrue ..... [L476](#), [L504](#), [L534](#)  
 \@ovdx ..... [L431](#), [L487](#),  
     [L489](#), [L495](#), [L497](#), [L516](#), [L518](#),  
     [L526](#), [L528](#), [L543](#), [L551](#), [L553](#),  
     [L589](#), [L592](#), [L599](#), [L601](#), [L693](#),  
     [L694](#), [L695](#), [L696](#), [L712](#), [L713](#),  
     [L714](#), [L717](#), [L733](#), [L753](#), [L754](#),  
     [L755](#), [L756](#), [L772](#), [L773](#), [L775](#), [L789](#)  
 \@ovdy ..... [L431](#), [L488](#),  
     [L490](#), [L496](#), [L497](#), [L517](#), [L519](#),  
     [L527](#), [L528](#), [L544](#), [L552](#), [L553](#),  
     [L564](#), [L569](#), [L577](#), [L581](#), [L700](#),  
     [L701](#), [L702](#), [L703](#), [L718](#), [L719](#),  
     [L720](#), [L723](#), [L737](#), [L760](#), [L761](#),  
     [L762](#), [L763](#), [L776](#), [L777](#), [L779](#), [L793](#)  
 \@ovhlinefalse ..... [L477](#), [L505](#)  
 \@ovhlinetrue ..... [L456](#),  
     [L460](#), [L465](#), [L483](#), [L489](#), [L511](#), [L518](#)  
 \@ovhorz ..... [L494](#),  
     [L495](#), [L525](#), [L526](#), [L550](#), [L551](#), [L584](#)  
 \@ovltrue ..... [L476](#), [L504](#), [L534](#)

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=lpageno.dtx, F=lxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

\@ovri ..... J33, L431, L486, L515, L542, L564, L577, L593, L602  
 \@ovro ... L431, L486, L495, L496, L515, L526, L527, L542, L551, L552, L563, L569, L576, L581, L588, L598, L614, L621, L630, L639  
 \@ovrtrue ..... L476, L504, L534  
 \@ovttrue ..... L476, L504, L534  
 \@ovvert ..... L492, L493, L521, L523, L546, L548, L557  
 \@ovvlinefalse ..... L477, L505  
 \@ovvlinetrue ..... L459, L482, L490, L510, L519  
 \@ovxx ..... L431, L480, L482, L483, L487, L493, L494, L508, L510, L511, L516, L523, L525, L537, L539, L543, L548, L550, L588, L598, L690, L691, L692, L696, L705, L706, L715, L716, L717, L732, L750, L751, L752, L756, L765, L766, L774, L775, L788  
 \@ovyy ..... L431, L481, L482, L483, L488, L495, L509, L510, L511, L517, L526, L538, L539, L544, L551, L561, L574, L697, L698, L699, L703, L705, L721, L722, L723, L736, L757, L758, L759, L763, L765, L778, L779, L792  
 \@p@filename ..... S90, S97, S105, S108, S115, S120  
 \@p@filepath ..... S91, S138, S147  
 \@p@filepath@aux ... S139, S140, S148  
 \@pagedp ..... V117, V308, V313, V1091, V1244, V1841, V1851  
 \@pageht ..... V116, V309, V313, V315, V316, V317, V321, V1090, V1243, V1824, V1831  
 \@par ..... m3, m5  
 \@parboxrestore ..... J266, J322, J347, J378, J396, J413, O19, O100, O169, O342, O360, O470, O489, O507, V219, V601, V659  
 \@parboxto ..... J261  
 \@parmmoderr ... l271, O58, O127, O316  
 \@parse@version ..... c80, c101, c102, c114, c115, c162, c163, c164, S195, S201, S202, S212, S1509, S1524, S1561, S1637, S1661  
 \@parse@version@ S189, S190, S195, S207  
 \@parse@version@dash .... S213, S215  
 \@partaux ..... q3, q222, q266, q284, q308, q310, q311, q331, q363, q365, q366, q372, q384, q386, q389  
 \@partlist ..... q231, q235, q236, q238, q262, q281, q304, q359  
 \@partswfalse ..... q6  
 \@partswtrue ..... q230, q260, q280  
 \@pass@ptions ..... S380, S382, S402, S403, S405, S418, S419, S421, S428, S429, S430, S890, S972  
 \@pboxswfalse ..... J264, J340  
 \@pboxswtrue ..... J274  
 \@openup ..... H186, H187  
 \@percentchar ..... a106, S1167, S1169, S1171, S1173, S1300, S1302, S1304, S1306, S1389, S1391, S1393, S1395, S1443  
 \@picbox ..... L6, L31, L43, L51, L52  
 \@picht ..... L6, L29, L42, L51  
 \@picture ..... L23, L24  
 \@picture@warn L223, L441, L445, L449  
 \@pkgextension ..... 813, S31, S155, S163, S219, S429, S619, S622, S661, S670, S741, S769, S796, S901, S928, S1058, T449, T455  
 \@plus ..... f26, o456, N16, N206, N229, R54, V2340, V2341, V2342, V2345, V2346, V2350, V2351, V2352, V2356, V2357, V2358  
 \@pnumwidth ..... N218, N241  
 \@popfilename ..... 791, 792, 811, S33, S876, S994  
 \@pr@videopackage ..... S293, S307, S310, S331, S333, S344, S346, S357  
 \@preamble ..... K190, K192, K200, K237, K256, K258, K259, K263, K278, K296, K297, K334  
 \@preamblecmds ..... 779, f66, q67, q135, q190, S1675, S1676  
 \@preamerr ... l260, K199, K274, K355  
 \@process@pti@ons ..... S488, S506, S523, S530, S531, S544, S548, S550  
 \@process@ptions ... S475, S477, S489  
 \@protected@testopt f89, f101, f580, f592  
 \@providesfile . a98, a99, S367, X627  
 \@optionlist ..... S152, S222, S474, S828, S834, S962, S968, S1059  
 \@pushfilename ..... 245, 791, 792, S33, S818, S949  
 \@put L452, L497, L528, L553, L621, L639  
 \@qend ..... f136, f671, l236  
 \@qrelax ..... f137, f671  
 \@raw@classoptionslist .... S11, S728  
 \@rc@ifdefinable . f130, f132, f238, r14  
 \@reargdef ..... f122  
 \@refundefined ..... q55, q124, q179, F3, G47, G87

**File Key:** a=ltirchck.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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lt pictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=lt hyphen.dtx, X=ltfinal.dtx

\@reinserts ..... V327, V330, V516  
 \@remove@eq@value ... S446, S511, S571  
 \@remove@tlig ..... r996, r1004  
 \@remove@tlig@ ..... r996, r997  
 \@remove@tlig@@ ..... r997, r1000  
 \@remove@tlig@@@ ..... r1030  
 \@remove@tlig@@@@ ..... r1010  
 \@removeelement ..... k32, S570, S579  
 \@removefromreset .....  
 ..... s46, s70, s71, s81, s104, s107  
 \@reqcolroom ..... V1090,  
 V1091, V1094, V1096, V1097,  
 V1102, V1106, V1108, V1136,  
 V1137, V1243, V1244, V1248,  
 V1251, V1252, V1257, V1264,  
 V1266, V1298, V1299, V1410,  
 V1412, V1414, V1417, V1419,  
 V1493, V1496, V1499, V1504,  
 V1506, V2020, V2137, V2142, V2145  
 \@reserveda ..... 439  
 \@reset@ptions .....  
 ..... S840, S877, S888, S953, S995, S1005  
 \@resetactivechars ..... V575, V598, V656  
 \@resetfps ..... V1205, V1374, V2087  
 \@restorepar .....  
 ..... m5, o341, o357, o375, o391, I127, I135  
 \@reversemarginfalse ..... O388, V101  
 \@reversemargintrue ..... O387  
 \@rightmark ..... R16, R51  
 \@rightskip .....  
 ..... G363, G382, G400, J295, J316, I75  
 \@rjfieldfalse ..... K34, K77  
 \@rjfieldtrue ..... K125  
 \@rmfamilyhook ..... z429, z451, z463, z490  
 \@roman ..... s137, s143  
 \@rsbox ..... J456, J463, J467  
 \@rtab ..... K71, K86  
 \@rule ..... J428, J433, J437  
 \@sanitize ..... f695, P7, P18, P24, P35  
 \@savebox ..... J122, J129, J135  
 \@savemarbox ..... O326, O327, O330, O333  
 \@savepicbox ..... J122, J129, J139  
 \@savsf ..... o123, o129,  
 o138, o157, o171, o183, o197, o211  
 \@savsk ..... o123, o128,  
 o139, o158, o172, o184, o198, o212  
 \@scolelt ..... V788, V853  
 \@sdblcolelt ..... V805, V825, V854  
 \@secCntformat ..... N60, N111  
 \@secondoftwo ..... a88,  
 e84, e36, e40, f212, f445, f539, f589,  
 f640, f647, f667, f714, q451, q496,  
 q513, r131, s190, s195, z512, D71,  
 ..... D838, D889, D905, F21, R17, S159,  
 S191, S203, S236, S254, S1666, X633  
 \@secPenalty ..... o17, N36, N50  
 \@sect ..... N54, N55  
 \@seqNcr ..... H412  
 \@set@curr@file@aux ..... T330  
 \@setckpt ..... q384, q391, G23, G76  
 \@setfloattypecounts ..... V1074, V1224,  
V1395, V1476, V1571, V1692, V2034  
 \@setfontsize ..... z637  
 \@setfps ..... O34  
 \@setfpsbit ..... O73, O75,  
 O77, O85, O143, O146, O149, V2078  
 \@setmarks ..... V2248, V2250, V2265  
 \@setminipage .....  
 ..... J352, O21, O177, O185, O376  
 \@setnobreak ..... O179, O375  
 \@setpar ..... m3, I78  
 \@setref ..... F10  
 \@setsize ..... z637  
 \@settab ..... K71, K93  
 \@settodim ..... t17  
 \@settopoint ..... t22  
 \@setupverbvisiblespace .....  
 ..... G473, G484, G496, G513, G528  
 \@sffamilyhook ..... z429, z456, z464, z491  
 \@sharp ..... K196, K235, K265,  
 K280, K281, K301, K303, K305, K333  
 \@shipoutsetup ..... V590  
 \@shortstack ..... L132, L133  
 \@show@... ..... 92  
 \@show@DeclareRobustCommand .....  
 ..... f507, f521, f566, f617, f620  
 \@show@newcommand .....  
 ..... 94, f508, f521, f574, f600, f617, f623  
 \@show@newcommand@aux ..... 95, f600, f627  
 \@showcommandlisthook .....  
 ..... 92, 138, 141, f504, f506, f514, g1223  
 \@skipping@modulefalse ..... c151, c170, c185  
 \@skipping@moduletrue ..... c150, c174  
 \@sline ..... L167, L179, L184, L269  
 \@slowromancap ..... s144, s145  
 \@spaces ..... l218  
 \@specialoutput ..... V256  
 \@specialpagefalse ..... V97, V607, V665  
 \@specialpagetrue ..... R9  
 \@specialstyle ..... R9, V607, V665  
 \@spToken ..... f680, f690  
 \@sqrt ..... H342  
 \@sect ..... N53, N112  
 \@stackcr ..... L139, L142  
 \@star@or@long ..... f72,  
 f77, f124, f146, f152, f178, f187, f221  
 \@startcolumn ..... V263, V270, V775

**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=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

```

\@startdblcolumn V775, V2270, V2273, V2310, V2316
\@startfield K28, K46, K92, K104, K125, K133
\@startline K20, K57, K62, K68, K83, K154
\@startpbox K193, K236, K266, K332, K384, K386
\@startsection N39
\@starttoc N149
\@stopfield K32, K48, K86, K93, K125, K127, K140, K142, K154
\@stopline K30, K56, K85
\@stpelt s20, s23
\@string@makeletter 99, f701
\@strip@args r76
\@strip@tex@ext 332, q226, q236, q238, q243, q273
\@strip@tex@aux q243, q274
\@svector L244, L259, L269
\@sverb . 626, G500, G563, G572, G575
\@svsec N57, N60, N66, N78
\@svsechd N76, N101, N121
\@swaptwoargs q530, q532, S850, T153, T167, T187
\@sxverbatim G408, G485, G492
\@tabacckludge r223, r225, r454, r455, D193, D200, D202
\@tabacol K178, K258
\@tabarray K170, K180, K181
\@tabclassiv K180, K330
\@tabclassz K179, K282
\@tabcr K56, K73
\@tabfbox K16, K80, K82
\@tablab K72, K126
\@tabminus K72, K117
\@tabplus K72, K110
\@tabpush K11, K77, K85, K140, K143, K145
\@tabrj K72, K124
\@tabular K174, K177, K178
\@tabularcr K180, K208
\@tempa F47, F48, F59, F60
\@tempboxa .. j13, r69, t17, t18, J29, J30, J31, J32, J37, J38, J39, J40, J175, J205, J212, J222, J343, J366, J473, J474, J475, J482, J483, J484, J485, L304, L305, L447, L448, L486, L491, L496, L497, L515, L520, L527, L528, L542, L545, L552, L553, L614, L615, L620, L621, L630, L631, L638, L639, L724, L742, L780, L798, N138, N139, O322, O380, V305, V377, V382, V383, V424, V429, V430, V566, V626, V633, V634, V685, V692, V693, V721, V725, V737, V743, V750, V751, V752, V753, V757, V765, I205, I211, I212, I214
\@tempcnta j7, y862, y863, y864, y865, y869, K242, K243, K244, K245, L187, L188, L214, L215, L216, L229, L230, L231, L232, L239, L240, L254, L255, L270, L271, L276, L278, L279, L280, L281, L282, L285, L287, L288, L289, L290, L291, L292, L293, L294, L295, L296, L335, L336, L337, L338, L339, L360, L361, L362, L363, L364, L365, L392, L393, L394, L395, L396, L414, L416, L418, L419, L421, L423, L438, L439, L440, L442, L444, L446, L448, L562, L567, L575, L579, L616, L617, L618, L619, L632, L633, L635, L636, L658, L659, L660, L661, L662, L663, L711, L731, L771, L787, O62, O68, O70, O79, O80, O90, O91, O131, O137, O139, O152, O153, O159, O160, S1115, S1117, S1118, S1119, S1246, S1248, S1249, S1250, S1360, S1362, S1363, S1364, V16, V18, V20, V934, V935, V936, V937, V957, V958, V959, V960, V982, V985, V1018, V1021, V1132, V1294, V1574, V1577, V1695, V1698, V1813, V1815, V1818, V1820, V1822, V1844, V2068, V2069, V2073, V2079, V2083, X213, X218, X219, X220, X350, X352, X353, X354, X361, X363, X364, X365, X371, X373, X374, X375, X382, X384, X385, X386, X412, X414, X415, X416, X438, X440, X441, X442, X448, X450, X451, X452, X481, X486, X487, X488
\@tempcntb j7, y863, y867, y869, L279, L280, L281, L283, L284, L285, L562, L563, L567, L568, L575, L576, L579, L580, O88, O89, O90, O157, O158, O159, V17, V20, V21, V2079, V2080, V2081, X214, X218, X482, X486
\@tempdim a j10, u219, u224, H173, H176, H182, J43, J44, J204, J205, J210, J211, J212, J214, J265, J266, J341

```

**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=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

J345, J440, J443, J444, J471, J473, J479, J482, K35, K36, K37, K88, K89, K90, K91, K218, K219, L210, L211, L213, L214, L215, L216, L217, L218, L437, L438, L439, L448, L487, L488, L492, L493, L516, L517, L521, L523, L543, L544, L546, L548, L617, L619, L634, L636, L637, L657, L658, L659, N211, N212, N234, N235, N248, O196, O198, O218, O220, O258, O259, O260, V229, V230, V231, V487, V489, V535, V537, V538, V543, V548, V552, V557, V561, V917, V920, V940, V950, V962, V972, V1637, V1638, V1641, V1642, V1762, V1763, V1767, V1768, V1823, V1824, V1825, V1826, V1829, V1832, V1835, V1837, V2186, V2187, V2189, V2190  
 \tempdimb ..... j10, u220, u225, u624, u628, w179, w180, w437, w460, w461, w470, w471, w475, w493, w496, w499, w501, J268, J269, J441, J444, J472, J474, J480, J483, L211, L212, L357, L359, L362, L365, L482, L484, L485, L510, L513, L514, L539, L540, L541, L612, L613, L622, L628, L629, L640, L648, L649, L654, V940, V941, V942, V943, V950, V962, V963, V964, V965, V972  
 \tempdimc ..... j10, w454, w455, w457, w458, w460, w461, J53, J54, J64, J65, J442, J443, J444, L30, L31, L32, L33, L34, L35, L60, L61, L63, L64, L332, L333, L334  
 \tempskipa ..... j14, o44, o47, o48, o285, o292, o294, o297, w181, w182, N42, N44, N45, N50, N62, N63, N88, N89, N91, N103, N104, N113, N114, V1872, V1873, V1875, V1883, I116, I117, I118, I150, I152, I153, I154, I222, I223, I224  
 \tempskipb ..... . 316, j14, o220, o222, o224, o227, o229, o243, o261, o283, o285, o286, o290, o292, o294, o295, o318, o321  
 \tempswafalse ..... a78, b262, q302, q357, r1529, u94, u673, y404, y459, y523, y604, y1124, y1130, G25, G78, G423, G444, Q13, S1092, S1108, S1223, S1239, T279, T300, V988, V1024, V1580, V1701  
 \tempswatrue ..... a79, b268, q300, q305, q355, q360, r1532, r1533, u97, u674, u675, u678, u681, y407, y462, y526, y607, y1087, G121, G428, G449, Q13, S1089, S1220, T279, T300, V1582, V1605, V1703, V1728, V2147, V2164  
 \temptokena ..... 613, j16, G133, G137, G146, G159, G160, R26, R27, R34, R35, R48, R49  
 \testdef ..... G24, G77, G119  
 \testfalse ..... V12, V14, V15  
 \testfp ..... V882, V902, V938, V961, V2071, V2200, V2217  
 \testopt ..... f33, f79, f99, f103, f148, o7, o8, o9, o10, H383  
 \testpatch ..... K270, K348  
 \testtrue ..... V13, V21, V356, V885, V904, V944, V966, V2075  
 \testwrongwidth ..... V345, V883, V939, V1112, V1426, V1631  
 \textcomposite ... r76, r1061, r1066  
 \textcomposite@x ..... r76  
 \textbottom ..... . R54, R56, V511, V549, V563, V572  
 \textfloatsheight ..... V476, V1087, V1089, V1139, V1140, V1145, V1240, V1242, V1302, V1304, V1310, V2020  
 \textmin ..... O285, O286, O299, O300, V112, V1089, V1093, V1096, V1097, V1242, V1247, V1251, V1252, V1414, V1499, V1598, V1600, V1616, V1720, V1722, V1740, V2128, V2130, V2132  
 \textsubscript ..... O424, O432  
 \textsuperscript .. O401, O403, O404  
 \texttop . R54, R56, V507, V530, V572  
 \tfor ..... k25, k26  
 \tfor ..... k25, q497, q514, q658, C88, J57, J76, K268, L478, L506, L535, O63, O132  
 \tforloop ..... k27, k28, k30  
 \thanks ..... N11, N34  
 \thefnmark J381, J398, J415, O400, O401, O453, O458, O473, O491, O509, O521, O526, O537, O542, O553  
 \thefoot ..... V124, V610, V613, V640, V669, V672, V699  
 \thehead ..... V123, V610, V612, V630, V669, V671, V689  
 \themargin ..... V74, V611, V613, V625, V670, V672, V684

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

\@themark R25, R26, R33, R34, R49, R52  
 \@thirdofthree ..... f216, r197  
 \@thm ..... M12, M18, M24, M26  
 \@thmcounter ..... M11, M17, M33  
 \@thmcountersep ..... M10, M33  
 \@title ..... N7, N31  
 \@tocrmarg ..... N207, N230  
 \@toodeep ..... I253, I36, I232, I243  
 \@toplist ..... V64, V384, V385,  
     V431, V432, V716, V722, V732,  
     V733, V1025, V1037, V1954, V1982  
 \@topnewpage ..... V199  
 \@topnum O271, V105, V1022, V1023,  
     V1037, V1041, V1458, V1463,  
     V1551, V1558, V1945, V1973, V2014  
 \@toproom ..... O273,  
     V106, V1025, V1037, V1946, V1974  
 \@topsep ..... I1, I71, I73, I171  
 \@topsepadd ..... I1, I59, I61, I71, I124  
 \@totallleftmargin 645, G419, G441,  
     J294, J315, K35, K76, K81, I9, I53, I54  
 \@trivlist ..... I48, I57, I92  
 \@tryfcolumn ..... V778,  
     V798, V816, V832, V2201, V2218  
 \@trylist V841, V844, V877, V897, V919  
 \@ttfamilyhook . z429, z461, z465, z492  
 \@twoclasseserror ..... S610, S1078  
 \@twocolumnfalse ..... V99, V147  
 \@twocolumntrue ..... V206  
 \@twoheadclasserror S875, S993, S1073  
 \@twosidefalse ..... V100  
 \@typein ..... f32, f33, f40, f48  
 \@typeset@protect ..... 84,  
     f102, f243, f250, f252, r26, r32, r210,  
     r218, z638, G193, G238, G256, X344  
 \@uclclist .. r1498, r1499, r1552, X551  
 \@undefined ..... L57, L71, L82, L91,  
     L162, L174, L237, L252, L313, L371  
 \@undefined ..... 855, a68, a69,  
     a108, a109, a110, a131, a139, a147,  
     a154, a205, a209, a235, a242, a326,  
     a327, b65, b81, b105, b106, b121,  
     b122, b127, b136, b149, b184, b189,  
     b222, b223, b246, b256, b291, b349,  
     b359, b361, b516, b584, b624, b639,  
     c50, c59, d2, d15, d16, d17, d29,  
     d30, d74, d84, d173, d181, d189,  
     d197, d229, d230, d231, d232, d233,  
     d234, d235, d236, d237, d238, d239,  
     d240, d241, d242, d243, d244, d245,  
     d246, d247, d253, e4, e13, e14,  
     e15, e16, e146, e147, e148, e149,  
     e150, e151, 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,  
     o89, o470, o476, q61, q62, q129,  
     q130, q184, q185, q273, q274, q422,  
     q423, q424, q429, q450, q471, q486,  
     q541, r195, r197, r333, r335, r337,  
     r339, r341, r343, r345, r347, r349,  
     r351, r370, r372, r374, r458, r700,  
     r703, s204, u434, u474, u536, u569,  
     u633, u640, v388, v411, v420, v509,  
     v510, v511, v512, v513, v514, v515,  
     v516, v517, v518, v529, v534, v539,  
     v603, v604, v605, v606, v607, v608,  
     v609, v634, v714, v716, v717, v719,  
     v720, v721, v723, w552, w553, x4,  
     x5, x6, x7, x8, x9, x10, x11, x12,  
     x13, x14, x15, x16, x17, x18, x19,  
     x20, y141, y720, y827, z35, z58,  
     z71, z91, z104, z105, z106, z107,  
     z108, z109, z110, z111, z112, z113,  
     z114, z116, z117, z118, z231, z232,  
     z233, z234, z235, z236, z237, z238,  
     z239, z240, z242, z243, z244, z304,  
     z305, z472, z490, z491, z492, z534,  
     z535, z536, z537, z538, z578, z579,  
     z580, z581, z582, z583, z594, z716,  
     A15, A50, A65, C37, C38, C39,  
     C122, D159, D367, D617, D620,  
     D621, D651, D652, D653, D656,  
     D657, D658, D659, D660, D661,  
     D666, D667, D668, D669, D674,  
     D675, D676, D677, D680, D681,  
     D682, D684, D685, D690, D691,  
     D692, D693, D694, D695, D696,  
     D697, D698, D699, D700, D701,  
     D702, D703, D704, D705, D707,  
     D708, D710, D711, D712, D713,  
     D714, D715, D716, D717, D718,  
     D719, D721, D722, D723, D724,  
     D725, D726, D727, D728, D729,  
     D731, D732, D733, D734, D735,  
     D736, D737, D738, D739, D740,  
     D741, D742, D743, D744, D745,  
     D746, D747, D748, D749, D750,  
     D751, D752, D753, D754, D755,  
     D756, D757, D762, D763, D765,  
     D766, D767, D768, D769, D770,  
     D771, D772, D774, D775, D777,  
     D778, D780, D781, D782, D783,  
     D785, D786, D787, D789, D791,

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

D792, D793, D794, D795, D796,  
 D797, D799, D800, D801, D802,  
 D803, D804, D805, D806, D807,  
 F53, F69, F70, G101, G153, G154,  
 G155, G156, G291, G292, G312,  
 G313, G314, G315, G467, G494,  
 G495, G496, G497, G530, H223,  
 H233, H234, H270, H273, H316,  
 H329, J21, J130, J194, J250, J434,  
 J464, L18, L467, L468, N197, N245,  
 O5, O429, O448, O562, Q33, S4,  
 S25, S146, S147, S148, S207, S353,  
 S461, S511, S880, S918, S920, S926,  
 S983, S998, S999, S1014, S1128,  
 S1206, S1209, S1259, S1338, S1341,  
 S1351, S1352, S1353, S1354, S1355,  
 S1356, S1357, S1428, S1431, S1447,  
 S1467, S1474, T18, T19, T20, T21,  
 T147, T189, T190, T197, T198,  
 T199, T327, T455, T461, T462,  
 T463, U427, U428, U429, U430,  
 U431, U433, U434, U435, U442,  
 U443, U445, U447, U448, U449,  
 U452, U475, V36, V368, V369,  
 V1924, V1925, V1926, V1927, X10,  
 X18, X25, X40, X59, X68, X75,  
 X84, X94, X146, X147, X256, X269,  
 X282, X302, X327, X337, X338,  
 X339, X368, X370, X409, X410,  
 X429, X430, X431, X432, X433,  
 X434, X435, X436, X437, X454,  
 X470, X471, X472, X520, X521,  
 X581, X616, X617, X618, X619, X620  
 \@unexpandable@protect . . . . .  
 ... f220, f255, f261, f266, q210, K264  
 \@unknownoptionerror . . . . .  
 ... S1009, S1048, S1061  
 \@unprocessedoptions . . . . .  
 ... 812, 814, S560, S669,  
 S880, S919, S920, S980, S984, S1063  
 \@unused . . . . .  
 ... b307, f10, f17, l15, l32, l59, S1452  
 \@unusedoptionlist . . . . .  
 ... q18, q20, q87, q89, q144, q146,  
 S12, S453, S454, S464, S465, S572, S580  
 \@upline . . . . . L297, L298, L304  
 \@upordown L195, L196, L204, L225, L273  
 \@upvector . . . . . L268, L304  
 \@use@option . . . . . S484, S500,  
 S518, S534, S536, S553, S555, S565  
 \@use@text@encoding r150, D16, D1141  
 \@vbsphack . . . . . o219  
 \@verb . . . . . G563, G572, G575  
 \@verbatim . . . . . G413, G459, G483, G492  
 \@verbvisiblespacebox . . . . .  
 ... G478, G479, G482, G497  
 \@verreq . . . . . A457, A458  
 \@viiipt . . . . . u716  
 \@viipt . . . . . u715  
 \@vipt . . . . . u714  
 \@vline . . . . . L166, L178, L297  
 \@vobeyspaces . . . . .  
 ... G406, G459, G485, G513, G528, G575  
 \@vpt . . . . . u713  
 \@vspace . . . . . o330  
 \@vspace@calcify . . . . .  
 ... o79, o101, o241, o337, o342, o352,  
 o360, G342, H391, K62, K224, L148  
 \@vspacer . . . . . o330  
 \@vtryfc . . . . . V847, V855  
 \@vvector . . . . . L243, L258, L268  
 \@warning . . . . . l215  
 \@wckptelt . . . . . q385, q388  
 \@whiledim . . . . . k7, L123, L203  
 \@whilenoop . . . . . k3  
 \@whilenum . . . . . k3, K244, L104,  
 L116, L336, L338, L361, L364,  
 L393, L395, L416, L421, L731, L787  
 \@whilesw . . . . . k10, V264, V394,  
 V403, V441, V451, V2271, V2311  
 \@whileswnoop . . . . . k10  
 \@wholewidth J160, J162, J163, J165,  
 J167, J168, J169, J170, L2, L126,  
 L129, L131, L299, L302, L350,  
 L359, L406, L413, L565, L578,  
 L590, L600, L679, L680, L728, L784  
 \@width . . . . .  
 ... b439, f26, o452, r289, r294, w192,  
 A618, J165, J167, J218, J225,  
 J444, J488, K188, K219, K347,  
 K366, L227, L299, L302, L325,  
 L333, L350, L359, L384, L392,  
 L406, L413, L565, L578, L728,  
 L784, O395, V1851, V2260, V2294  
 \@wglossary . . . . . P25, P30  
 \@wrindex . . . . . P8, P13  
 \@writeckpt . . . . . q329, q370, q382  
 \@writefile . . . . .  
 ... 613, 746, q32, q101, q159, G140, N183  
 \@writesetup . . . . . V590  
 \@wrong@font@char . . . . .  
 ... r161, u537, u571, u584  
 \@wtryfc . . . . . V857, V867  
 \@x@protect . . . . . f105, f242  
 \@x@sf . . . . . O531, O533  
 \@xDeclareMathDelimiter . . . y892, y948  
 \@xaddvskip . . . . . o219, o244, o262

**File Key:** a=ltdefchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxexpl.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=ltspacedtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

\@xarg ..... L163, L166, L175, L178, L185, L189, L190, L226, L228, L238, L239, L243, L253, L254, L258, L266, L274, L664  
\@xargarraycr ..... K205, K214, K218  
\@xargdef ..... f80  
\@xarraycr ..... K202, K203  
\@xbitor ..... V15, V17  
\@xcentercr ..... G325, G332, G336  
\@xdblarg ..... f693  
\@xdblfloat ..... O264  
\@xdim .....  
L84, L93, L105, L107, L117, L119, L668, L732, L733, L734, L735, L741, L788, L789, L790, L791, L797  
\@xeqnacr ..... H375  
\@xexnoop ..... K238, K248  
\@xexpast ..... K239, K240  
\@xfloat ..... O28, O29, O34, O266  
\@xfootnote ..... O452, O455  
\@xfootnotemark ..... O519, O523  
\@xfootnotenext ..... O536, O539  
\@xhline ..... K360, K361  
\@xifnch ..... f681, f691  
\@xiipt ..... u720, A164, A166, A167  
\@xipt ..... u719, A163  
\@xivpt ..... u721, A165, A167  
\@xmpar ..... O324, O325  
\@xnewline ..... o60, o61, o72, o73, o93  
\@xnnext ..... V10, V11  
\@xnthm ..... M5, M6  
\@xobeysp .....  
. 320, o420, G407, G408, G475, G479  
\@xprocess@ptions .....  
. S475, S492, S494, S510, S512, S526  
\@xpt ..... u718, A162, A165, A166  
\@xsect ..... N86, N87, N123  
\@xtabcr ..... K56, K57  
\@xtabularcr ..... K209, K210  
\@xthm ..... M28, M29  
\@xtryfc ..... V844, V872  
\@xtypein ..... f33, f35, f42  
\@xverbatim ..... G408, G459  
\@xvipt ..... u722, A166, A168  
\@xxDeclareMathDelimiter ..... y877, y881  
\@xxpt ..... u723, A167, A168  
\@xxvpt ..... u724, A168  
\@xxxii ..... j2, r425, r427,  
O89, O158, V879, V880, V899,  
V900, V935, V936, V958, V959, V2037  
\@xympar ..... O328, O332, O378  
\@yarg ..... L163, L167, L175, L179, L185, L186, L195, L238, L244, L253, L259, L268, L270, L297, L664  
\@yargarraycr ..... K206, K216, K220  
\@yargd@f ..... f107  
\@yargdef ..... f84, f94, f107, f123  
\@ydim .....  
L85, L94, L105, L108, L117, L119, L668, L736, L737, L738, L739, L740, L792, L793, L794, L795, L796  
\@yeqnacr ..... H375  
\@ympar ..... O324, O329  
\@ynthm ..... M5, M14  
\@ythm ..... M28, M29  
\@ytryfc ..... V890, V909, V913  
\@yyarg .....  
. L185, L186, L187, L190, L274, L664  
\@ztryfc ..... V918, V929  
\@zend ..... 198  
\@accent@spacefactor ..... r70, r73, r74  
\@active@math@prime ..... H240, H241, V588  
\@add@accent ..... r65, r67  
\@add@percent@to@temptokena .....  
. 614, G127, G143, G145, G154  
\@add@unicode@accent ..... r1040, r1044  
\@addto@hook ..... u152, u154, u712, y386, y482, y486, y503, y627, y633, y641, y657, y660, y663, y1096, y1103, y1106  
\AddToHook ..... 250  
\AddToHookNext ..... 250  
\alloc@ ..... b90, b91, b92, b93, b94, b95, b96, b97, b98, b99, b226, d22, d26, d38, u14  
\alpha@elt .....  
. y45, y390, y577, y679, y1095, y1096  
\alpha@list .....  
. y41, y43, y399, y565, y577, y622, y677, y678, y1091, y1097, y1098  
\apptocmd ..... 250  
\atveryend@DEPRECATED ..... T543, T545  
\best@size ..... w438, w462, w468, w474  
\bf@def@ult ..... z397  
\bfdef@ult ..... 441, 518, 527, z200, z271, z272, z273, z314, z315, z316, z401, z442  
\bfdefault@previous ..... 518, z267, z270, z310, z313, A106, A116  
\bfseries@.. ..... 513  
\bfseries@rm .....  
. 513, 516, 527, z105, z128, z207, z210, z231, z258, z271, z314, z319, z355  
\bfseries@rm@kernel .....  
. 516, z108, z131, z207, z234  
\bfseries@sf .....  
. 510, 516, z106, z128, z211, z214, z232, z259, z272, z315, z320, z356  
\bfseries@sf@kernel .....  
. z109, z132, z211, z235

**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=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspacedtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

\bfseries@tt	516, z107, z128, z215, z218, z233, z260, z273, z316, z321, z357	
\bfseries@tt@kernel		
\bm@b	J37	
\bm@c	J37	
\bm@l	J37	
\bm@r	J37	
\bm@s	J37	
\bm@t	J37	
\bx@	942	
\bx@A	V30, V57	
\bx@AA	V40	
\bx@B	V30, V57	
\bx@BB	V40	
\bx@C	V30, V57	
\bx@CC	V40	
\bx@D	V30, V57	
\bx@DD	V40	
\bx@E	V30, V57	
\bx@EE	V40	
\bx@F	V31, V58	
\bx@FF	V41	
\bx@G	V31, V58	
\bx@GG	V41	
\bx@H	V31, V58	
\bx@HH	V41	
\bx@I	V31, V58	
\bx@II	V41	
\bx@J	V31, V58	
\bx@JJ	V41	
\bx@K	V32, V59	
\bx@KK	V42	
\bx@L	V32, V59	
\bx@LL	V42	
\bx@M	V32, V59	
\bx@MM	V42	
\bx@N	V32, V59	
\bx@NN	V42	
\bx@O	V33, V60	
\bx@OO	V43	
\bx@P	V33, V60	
\bx@PP	V43	
\bx@Q	V33, V60	
\bx@QQ	V43	
\bx@R	V33, V60	
\bx@RR	V43	
\bx@S	V38	
\bx@SS	V44	
\bx@T	V38	
\bx@TT	V44	
\bx@U	V38	
\bx@UU	V44	
\bx@V	V38	
\bx@VV	V44	
\bx@W	V39	
\bx@WW	V45	
\bx@X	V39	
\bx@XX	V45	
\bx@Y	V39	
\bx@YY	V45	
\bx@Z	V39	
\bx@ZZ	V25, V45, V55	
\c@bottomnumber	O269, O274, V2326	
\c@dbltopnumber		
	O268, O283, O297, V2333	
\c@enumi	I227	
\c@enumii	I227	
\c@enumiv	I227	
\c@equation	H336, H370, H513	
\c@errorcontextlines	I212	
\c@footnote	N12, O397, O525	
\c@localalphabets	y137	
\c@localmathalphabets		
	484, y141, y142, y150, y183	
\c@mpfootnote	J348, O399	
\c@ncel	A461, A462	
\c@page	E3, E6, E7, V138, V1818	
\c@sectiondepth	N56, N71, N81, N140	
\c@tocdepth	N140, N205, N228	
\c@topnumber	O267, O271, V2322	
\c@totalnumber	O270, O276, V2329	
\c@totalpages	U346, U430	
\calculate@math@sizes	u620, w219	
\catcodetable@atletter	d93, d238	
\catcodetable@initex	d93, d235	
\catcodetable@lateX	d93, d237	
\catcodetable@string	d93, d236	
\cdp@elt	u96, u116, u127, u128, u149, u152, u154, y311, y406, y461, y525, y606, X458, X459	
\cdp@list		
	u98, u114, u128, u156, u157, y329, y408, y463, y527, y608, X459	
\cf@encoding	r34, r41, r44, r51, r154, u256, u266, u276, u326	
\ch@ck	b206, b207, b208, b209, b236, b248, b249, b250, b251, b279, b281, b293, b294, b295, b296, b302, S1133, S1152, S1264, S1285, S1377	
\char@if@alph	f701	
\chardef@text@cmd	r3	
\check@command	f187, f189	
\check@icl		
	C9, C44, C49, C55, C63, C70, C72	
\check@icr		
	C9, C44, C50, C56, C64, C73, C78	

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

```

\check@mathfonts 485, p5, r302, r328, r360,
r1246, u348, u350, w250, y188, D672
\check@nocorr@ C46
\check@range w379, w380
\check@singl w378, w400
\cl@ckpt q385, s35
\cl@page E4
\col@number ... V95, V148, V208, V220
\color@beginninggroup
... u643, u703, H104, H134, J29,
J89, J176, J344, J382, J399, J416,
K47, K51, O475, O493, O511, V491
\color@endbox ... J89, O253, O348,
O366, V224, V631, V641, V690, V700
\color@endgroup
u648, u709, H104, H134, J29, J89,
J134, J155, J178, J364, J386, J403,
J419, K49, O479, O497, O514, V495
\color@hbox
... J89, V628, V638, V687, V697
\color@setgroup J89, J134, J153
\color@vbox
J89,
O96, O165, O339, O357, O381, V215
\conditionally@traceoff
... 285, g217, g229, l289
\conditionally@traceon ... g253, l289
\contionally@traceon 285
\copy@kernel@robust@command f541, f625
\count@ a66, a179, a180, a181,
a186, b41, b191, b192, b197, b199,
b205, b206, b207, b208, b209, b210,
b439, b440, c14, c15, c16, c17, c18,
c20, f169, f173, f275, f300, u677,
u683, u685, w22, w302, w304, w326,
w327, y383, y385, y389, y708, y709,
y710, y756, y757, y758, y817, y818,
y819, y865, y866, y867, y905, y906,
y907, y913, y914, y915, y959, y960,
y961, y967, y968, y969, y1028,
y1029, y1030, y1036, y1037, y1038,
C115, C118, L730, L731, L732,
L735, L736, L739, L743, L786,
L787, L788, L791, L792, L795,
L799, S1176, S1178, S1179, S1180,
S1309, S1311, S1312, S1313, S1398,
S1400, S1401, S1402, X225, X226,
X233, X235, X525, X526, X533, X535
\counterwithin@s s113, s114, s132
\counterwithin@x s113, s116, s133
\counterwithout@s ... s102, s103, s129
\counterwithout@x ... s102, s105, s130
\curr@fontshape . r180, u88, u363,
u371, u375, u377, u519, u525, u528,
u537, u544, u546, u554, u560, u563,
u571, u578, u580, v454, w92, w100,
w142, w169, w477, w497, w529,
w545, w560, y333, y338, z556, z565
\curr@math@size
... u352, w256, w262, w267, w284
\declare@commandcopy
... 91, f464, f468, f473, f476, f493
\declare@commandcopy@let
... 91, f481, f485, f495, f563, f564
\declare@file@substitution
... 839, T247, T530, U506
\declare@robustcommand
... f221
\DeclareEncodingSubset@aux
... D41, D43, D59
\DeclareFontEncoding@
... u122, u124, u139, X369, X389, X455
\DeclareFontEncoding@saved
... X369, X389, X471
\DeclareFontShape@
... u21, u22
\DeclareRobustCommand
... 251, 258
\DeclareSymbolFontAlphabet@
... y1083, y1086
\def
... 251, 253, 265, 266
\default@ds
... S442, S471, S535, S554, S1007, S1009
\default@family
... u129, u161,
u487, u501, u504, u529, u564, X460
\default@OM u136, u176, u179, u183, X467
\default@mextra
... x10, x89
\default@series
... u129, u162,
u488, u502, u505, u526, u561, X460
\default@shape
... u130, u163,
u489, u503, u506, u524, u559, X461
\default@T
... u170, u173, u183, u272
\define@mathalphabet
... x18, x131
\define@mathgroup
... x19, x135
\define@newfont
... u355, u364
\delayed@f@adjustment . 410, 437,
454, 532, u290, v395, v396, v397,
v399, v400, v411, v616, v617, v619,
v620, w122, w126, w134, w138, z662
\delayed@merge@font@series
... v396, v460, v475, v514, w133, w136
\delayed@merge@font@shape
... v617, v666, v721, w132, w135
\development@branch@name
... c11, c36, c50, c51, c52, c59, c60, c61
\dimen@
... b41, b436,
b437, b473, b474, b476, b478, l28,
l29, o349, o354, o383, o388, r429,
r430, r432, r433, r796, r797, u214,
u216, u222, u235, u238, u242, u623,
u624, u625, u629, w451, w452,

```

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
X=ltfinal.dtx

w453, w454, w458, D102, D104, D931, D933, H72, H73, H186, H187, H188, H189, J481, J484, K176, K177, V508, V510, V531, V533  
 \dimen@i ..... b41  
 \dimen@ii ..... b41, u218, u223  
 \disable@package@load ..... 839, 853, T444, U464  
 \displ@y ..... H191, H195, H196  
 \do@add@percent@to@temptokena ...  
     ..... G131, G137, G155  
 \do@emfont@update ... z557, z561, z581  
 \do@noligs ..... G577, G582  
 \do@subst@correction u84, w482, w537  
 \document@default@language ...  
     ... q51, q52, q120, q121, V597, X272  
 \document@select@group ...  
     ..... 484, y139, y145, y346  
 \dont@add@percent@to@temptokena ...  
     ..... G130, G132, G156  
 \dorestore@version ... y114, y119  
 \ds@ ..... S473, S1011  
 \dt@pfalse ..... H192  
 \dt@ptrue ..... H191  
 \e@alloc ... b51, b52, b53,  
     b55, b56, b63, b64, b66, b68, b79,  
     b82, b84, b138, b230, d15, d49, d79,  
     d89, d178, d186, d194, d202, X12, X33  
 \e@alloc@attribute@count ...  
     ..... d66, d74, d75, d76, d80, d229  
 \e@alloc@bytecode@count ...  
     ... d70, d189, d190, d191, d195, d245  
 \e@alloc@ccodetable@count ...  
     ..... d67, d84, d85, d86, d90, d233  
 \e@alloc@chardef . b60, b102, b210,  
     b211, d48, d178, d186, d194, d202, X12  
 \e@alloc@intercharclass@top ... X21  
 \e@alloc@luachunk@count ...  
     ... d71, d197, d198, d199, d203, d247  
 \e@alloc@luafunction@count . d68,  
     d173, d174, d175, d179, d239, d241  
 \e@alloc@top ...  
     ..... b55, b63, b102, b188, b259,  
     d47, d80, d179, d187, d195, d203, X12  
 \e@alloc@whatsit@count ...  
     ... d69, d181, d182, d183, d187, d243  
 \e@ch@ck ... b142, b152, d51, d55  
 \e@insert@top . b257, b259, b276, b291  
 \e@mathgroup@top ... 484,  
     b79, b124, y56, y149, y150, y224, y254  
 \em@currfont ... 529, z556, z567  
 \em@force .. 529, z565, z570, z573, z583  
 \emfontdeclare@clist ... 528,  
     z546, z548, z552, z557, z562, z568, z579  
 \empty@sfcnt .....  
     w490, w491, w492, w506, w511, w563  
 \enc@update .....  
     . u257, u259, u275, u278, w147, w173  
 \end@dblfloat ..... O205  
 \end@float 764, O189, O227, O243, O383  
 \endlinechar ..... 266  
 \err@rel@i ... x12, x99, x132, x136  
 \error@fontshape ..... u482,  
     u497, u522, u557, w107, w527, y332  
 \escapechar ..... 261  
 \et@xmaxfam ... d22, d26, d30, d38  
 \et@xmaxregs ...  
     d29, d31, d32, d33, d34, d35, d36, d37  
 \every@math@size ... u78, w235, w247  
 \execute@size@function ...  
     ..... w362, w390, w404, w421  
 \expand@font@defaults ...  
     ..... 199, z37, z148, z256, z279,  
     z309, z330, z354, z365, z396, z397,  
     z436, z438, z470, z472, z501, D26, D83  
 expand@font@defaults ... z420  
 \external@font ...  
     w84, w87, w98, w102, w104, w391,  
     w405, w467, w501, w569, w571, w573  
 \extra@def ..... x9, x84  
 \extract@alph@from@version ...  
     .. u597, u603, y162, y230, y260, y292  
 \extract@default@composite ...  
     ..... r1053, r1060  
 \extract@default@composite@a ...  
     ..... r1062, r1066  
 \extract@default@composite@b ...  
     ..... r1064, r1068  
 \extract@font ... u378, w81  
 \extract@fontinfo ... w358, w365  
 \extract@rangefontinfo ...  
     ..... w375, w382, w401, w434  
 \extract@sizefn ... w350, w372  
 \f@baselineskip ... r870, r1212,  
     u317, u324, u508, w121, w167,  
     w182, w186, w201, w215, w226, w240  
 \f@depth ..... O291, V345  
 \f@encoding ...  
     d252, d267, d275, r178, u251, u270,  
     u273, u274, u276, u326, u358, u363,  
     u382, u384, u386, u391, u393, u424,  
     u486, u518, u553, v439, v443, v652,  
     v656, w91, w128, w307, w517, y317  
 \f@family 410, 518, 526, u279, u287,  
     u299, u309, u320, u359, u363, u382,  
     u384, u386, u391, u393, u425, u504,  
     u529, u564, v439, v443, v652, v656,  
     w91, w128, y317, y348, z151, z178,

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lpictur.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

z179, z258, z259, z260, z281, z282,  
 z283, z319, z320, z321, z339, z340,  
 z341, z355, z356, z357, z366, z367,  
 z368, z504, z517, z659, z677, z693,  
 D23, D27, D29, D31, D63, D66,  
 D80, D84, D86, D88, D93, D100,  
 D897, D900, D914, D923, D929, D1144  
 $\backslash f@linespread$  .....  
     ..... u320, w120, w166, w183,  
     w184, w187, w195, w198, w209, w212  
 $\backslash f@series$  .. 428, 437, 438, 440, 444,  
     453, 454, 515, 517, 523, 525, 527,  
     534, p14, u279, u310, u321, u360,  
     u363, u505, u526, u561, v400, v410,  
     v419, v429, v463, v482, v483, v652,  
     v656, w125, w128, w131, y349,  
     z144, z160, z162, z166, z167, z168,  
     z189, z195, z198, z200, z222, z520,  
     z524, z660, z678, z694, z739, D7, D613  
 $\backslash f@series@saved$  ..... w125, w131  
 $\backslash f@shape$  .....  
     . 445, 453, 534, u279, u289, u301,  
     u311, u322, u361, u363, u506, u524,  
     u559, v439, v443, v620, v627, v633,  
     v643, v650, v654, v657, v660, v669,  
     v676, v678, v713, w124, w128,  
     w130, y350, z661, z679, z695, z740  
 $\backslash f@shape@saved$  ..... w124, w130  
 $\backslash f@size$  r180, r869, r1211, u88, u317,  
     u323, u362, u507, u546, u580, u622,  
     u623, u626, u627, w121, w142,  
     w167, w169, w180, w200, w215,  
     w218, w221, w226, w233, w240,  
     w252, w255, w261, w267, w284,  
     w285, w288, w293, w359, w366,  
     w385, w387, w402, w453, w455,  
     w457, w473, w474, w479, w493,  
     w505, w510, w522, w530, w535,  
     w561, w575, z556, z565, D102, D931  
 $\backslash f@user@size$  . w473, w478, w522, w535  
 $\backslash famdef@ult$  ..... z444  
 $\backslash filec@ntents$  .....  
     . S1087, S1090, S1093, S1104,  
     S1125, S1218, S1221, S1224, S1235,  
     S1256, S1349, S1371, S1459, S1674  
 $\backslash filec@ntents@checkdir$  .....  
     ..... S1110, S1112,  
     S1126, S1241, S1243, S1257, S1356  
 $\backslash filec@ntents@force$  .....  
     ..... S1106, S1237, S1352  
 $\backslash filec@ntents@noheader$  .....  
     ..... S1108, S1239, S1354  
 $\backslash filec@ntents@nosearch$  .....  
     ..... S1109, S1240, S1355  
  
 $\backslash filec@ntents@opt$  .. S1090, S1093,  
     S1095, S1221, S1224, S1226, S1351  
 $\backslash filec@ntents@OPTION$  ..... 818  
 $\backslash filec@ntents@overwrite$  .....  
     ..... S1107, S1238, S1353  
 $\backslash filec@ntents@where$  S1111, S1113,  
     S1138, S1242, S1244, S1269, S1357  
 $\backslash filename@area$  ..... a246,  
     a252, a259, a265, a272, a278, a285,  
     q553, q568, q578, q610, q611, q615,  
     q640, q643, q660, q672, q674, S857  
 $\backslash filename@base$  ..... a295, a302,  
     a315, q553, q568, q578, q610, q612,  
     q615, q640, q643, q667, q672, S858  
 $\backslash filename@dot$  ..... a313, a319  
 $\backslash filename@dots$  ..... a297, a299, a304  
 $\backslash filename@ext$  ..... 845,  
     a294, a301, a311, a313, q554, q569,  
     q579, q606, q607, q610, q613, q615,  
     q636, q637, q640, q643, q668, S859  
 $\backslash filename@parse$  a110, a242, q551,  
     q566, q576, q605, q635, q665, S856  
 $\backslash filename@path$  .. a247, a248, a253,  
     a260, a261, a266, a273, a274, a279  
 $\backslash filename@simple$  ..... a250, a263,  
     a276, a286, a290, a292, a307, a309  
 $\backslash finish@module@release$  ..... c86, c90  
 $\backslash finph@nt$  .....  
     H104, H106, H110, H111, H119, H120  
 $\backslash finsm@sh$  .....  
     H134, H136, H140, H141, H145, H146  
 $\backslash fix@penalty$  ..... C101  
 $\backslash fixed@sfcnt$  ..... w565, w566, w567  
 $\backslash fl@ShowFloat$  .. V1904, V1910, V1925  
 $\backslash fl@trace$  ..... V240, V267, V323,  
     V351, V358, V379, V426, V472,  
     V525, V540, V541, V542, V543,  
     V554, V555, V556, V557, V558,  
     V568, V781, V800, V819, V837,  
     V839, V978, V982, V994, V995,  
     V996, V997, V1003, V1006, V1014,  
     V1018, V1029, V1034, V1039,  
     V1040, V1041, V1042, V1049,  
     V1052, V1060, V1071, V1077,  
     V1082, V1087, V1093, V1094,  
     V1099, V1104, V1105, V1106,  
     V1114, V1118, V1123, V1127,  
     V1132, V1143, V1144, V1146,  
     V1164, V1173, V1179, V1188,  
     V1191, V1197, V1207, V1211,  
     V1221, V1227, V1233, V1239,  
     V1246, V1248, V1254, V1259,  
     V1261, V1263, V1271, V1276,  
     V1282, V1287, V1293, V1307,

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lpictur.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

V1308, V1311, V1332, V1341,  
 V1347, V1356, V1359, V1366,  
 V1376, V1380, V1392, V1398,  
 V1403, V1408, V1412, V1416,  
 V1417, V1424, V1429, V1433,  
 V1440, V1449, V1453, V1457,  
 V1458, V1462, V1463, V1473,  
 V1479, V1485, V1491, V1495,  
 V1501, V1503, V1511, V1516,  
 V1521, V1529, V1538, V1543,  
 V1548, V1550, V1555, V1557,  
 V1568, V1574, V1584, V1590,  
 V1594, V1595, V1600, V1601,  
 V1607, V1610, V1611, V1612,  
 V1619, V1620, V1621, V1629,  
 V1634, V1646, V1647, V1654,  
 V1657, V1665, V1669, V1673,  
 V1674, V1678, V1679, V1689,  
 V1695, V1705, V1711, V1715,  
 V1716, V1722, V1723, V1730,  
 V1733, V1734, V1735, V1743,  
 V1744, V1745, V1754, V1759,  
 V1772, V1774, V1781, V1784,  
 V1793, V1797, V1801, V1802,  
 V1806, V1807, V1859, V1864,  
 V1870, V1880, V1887, V1901,  
 V1902, V1906, V1917, V1929,  
 V2027, V2040, V2041, V2045,  
 V2048, V2050, V2053, V2056,  
 V2058, V2099, V2106, V2111,  
 V2117, V2122, V2126, V2132,  
 V2140, V2142, V2149, V2154,  
 V2159, V2161, V2167, V2169,  
 V2176, V2205, V2207, V2222,  
 V2224, V2238, V2263, V2267,  
 V2272, V2284, V2301, V2306, V2314  
 $\text{\f@tracemessage}$  . . . . .  
     . . . . . V1901, V1918, V1927, V1929  
 $\text{\f@traceval}$  . . . . . V1911, V1912,  
     V1913, V1914, V1917, V1926, V1929  
 $\text{\float@count}$  . . . . .  
     . . . . . b51, b52, b53, b62, b188,  
     b205, b210, b212, b213, b222, b230  
 $\text{\fmtversion@topatch}$  . . . . .  
     X578, X580, X592, X593, X605, X613  
 $\text{\font@info}$  . . . . . w99, w365, w434, w439  
 $\text{\font@name}$  . . . . . r179,  
     r182, u86, u194, u196, u354, u369,  
     u545, u579, w84, w88, w90, w105,  
     w141, w144, w154, w168, w171,  
     w330, w331, w332, w333, w334, w339  
 $\text{\font@submax}$  . . . . . w441, w470, w471,  
     G42, G44, G82, G84, X285, X287, X296  
 $\text{\fps@dbl}$  . . . . . O34  
 $\text{\freeze@math@version}$  . . . . . y154, y178  
 $\text{\frozen@everydisplay}$  . . . . . u344, u350  
 $\text{\frozen@everymath}$  . . . . . u344, u348  
 $\text{\g@addto@macro}$  . . . . . 205,  
     f813, i60, v731, S133, S397, S1019,  
     S1035, S1036, U350, U356, U397  
 $\text{\G@refundefinedfalse}$  . . . . . 604, F5  
 $\text{\G@refundefinedtrue}$  . . . . .  
     . . . . . F3, F12, Q21, Q48, Q65  
 $\text{\gen@sfcnt}$  . . . . . w502, w503, w504  
 $\text{\genb@sfcnt}$  . . . . . w507, w508, w509  
 $\text{\genb@x}$  . . . . . w510, w512  
 $\text{\genb@y}$  . . . . . w512  
 $\text{\get@cdp}$  . . . . . y479, y487, y520  
 $\text{\get@external@font}$  .. . w83, w96, w536  
 $\text{\getanddefine@fonts}$  . . . . .  
     . . . . . 486, u592, u610, w320, y59,  
     y87, y132, y159, y201, y227, y257,  
     y288, y386, y450, y484, y486, y503,  
     y626, y627, y659, y660, y1102, y1103  
 $\text{\glb@currsize}$  . . . . . q41, q110, q168,  
     u341, w217, w252, w256, w262, w285  
 $\text{\glb@settings}$  u342, w217, w264, w295  
 $\text{\gobble@finish@module@release}$  . . . . .  
     . . . . . c106, c108, c167  
 $\text{\group@elt}$  . . . . . y35,  
     y384, y421, y422, y443, y447, y1134  
 $\text{\group@list}$  . . . . .  
     . . . . . y388, y428, y441, y446, y447,  
     y476, y702, y750, y811, y896, y899,  
     y950, y953, y1019, y1022, y1089, y1140  
 $\text{\h@false}$  . . . . . H81  
 $\text{\h@true}$  . . . . . H82, H83  
 $\text{\hb@xt@}$  . . . . . b483, f29, r425,  
     H197, H363, H425, H440, H452,  
     H479, H510, J44, J65, J83, J205,  
     J489, J493, J494, J495, K37, L31,  
     L43, L62, L74, L105, L117, L265,  
     L299, L302, L305, L307, L314,  
     L373, L452, L588, L598, L741,  
     L797, N218, N241, N248, V630,  
     V640, V689, V699, V1843, V2257,  
     V2258, V2262, V2289, V2290, V2296  
 $\text{\hexnumber@}$  . . . . .  
     . . . . . y723, y731, y766, y774, y795,  
     y805, y831, y839, y847, y856, y859,  
     y868, y869, y908, y916, y962, y970,  
     y989, y990, y1000, y1001, y1006,  
     y1032, y1040, y1045, y1047, z645  
 $\text{\hgl@}$  . . . . . b438, b439  
 $\text{\hmode@bgroup}$  . . . . . r67, r75,  
     r327, r356, r390, r396, r427, r438,  
     r445, r476, r483, r486, r488, r496,

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

r512, r730, r760, r766, r798, r805,  
 r833, r836, r893, r1245, C7, D625, D632  
 \hmode@start@before@group .....  
     ..... r68, r151, r153, r159, r184  
 \if@afterindent ..... N124, N131  
 \if@compatibility ..... S2, S611  
 \if@endpe G214, G221, G273, G288, I138  
 \if@eqnsw ..... H344, H410  
 \if@fcolmade .....  
     ... V95, V264, V394, V403, V441,  
     V451, V779, V799, V817, V846,  
     V926, V2204, V2221, V2271, V2311  
 \if@filesw .....  
     ... q5, q36, q105,  
     q163, q297, q309, q330, q352, q364,  
     q371, q383, G21, G48, G74, G88,  
     N153, Q4, Q8, Q19, Q28, Q36, Q47,  
     Q64, S1136, S1153, S1267, S1286, U359  
 \if@firstamp ..... K251  
 \if@firstcolumn V95, V246, V279,  
     V396, V444, V1815, V2235, V2280  
 \if@font@series@context .....  
     ... 526, z509, z528, z536  
 \if@forced@series ..... 438, v401, z143  
 \if@ignore G4, G216, G223, G274, G289  
 \if@includeinrelease .....  
     ... c70, c73, c129, f767  
 \if@inlabel .....  
     V161, V188, I28, I65, I102, I160, I183  
 \if@insert .....  
     V95, V1057, V1169, V1203, V1337,  
     V1372, V1446, V1535, V1662, V1790  
 \if@minipage o239, o257, o276, o311,  
     G418, G440, J323, K79, O20, I149  
 \if@mparswitch ..... V95, V1817  
 \if@multiplelabels ..... F31  
 \if@negarg .... L157, L198, L212, L273  
 \if@newlist G460, V599, V644, V657,  
     V703, I29, I33, I69, I78, I106, I166  
 \if@nmbrlist ..... I33, I201  
 \if@no@font@opt ..... x16, x110, x129  
 \if@nobreak o120, o147, o278, o313,  
     q202, q214, J286, J307, N47, N128,  
     O180, O373, R29, R37, V165,  
     V192, V335, V1148, V1314, I167, I192  
 \if@noitemarg ..... I32, I199  
 \if@noparitem ..... I30, I157  
 \if@noparlist ..... I31, I114  
 \if@noskipsec .. o147, J287, J308,  
     N38, N40, N97, O374, V155, V182, I58  
 \if@ovb .....  
     . L427, L495, L526, L551, L562, L575  
 \if@ovhline ..... L459, L590  
 \if@owl .....  
     . L427, L493, L522, L547, L592, L601  
 \if@ovr .....  
     . L427, L492, L521, L546, L589, L599  
 \if@ovt .....  
     . L427, L494, L525, L550, L567, L579  
 \if@ovvline ..... L459, L565  
 \if@partsw ..... q5, q301, q356  
 \if@pboxsw ..... J278, J423  
 \if@reversemargin ..... V101, V1820  
 \if@reversemarginpar ..... V95  
 \if@rjfield ..... K19, K33  
 \if@skipping@module . c134, c146, c149  
 \if@specialpage ... V95, V606, V664  
 \if@tempswa ... a78, a79, a80, b270,  
     j9, q307, q362, r1534, u99, u687,  
     y409, y464, y528, y609, y1133, G50,  
     G90, G425, G446, Q75, S1165,  
     S1298, S1387, T285, T286, T306,  
     T307, V990, V1026, V1626, V1751  
 \if@test ..... V12,  
     V13, V887, V906, V946, V968,  
     V1032, V1116, V1125, V1274,  
     V1285, V1427, V1514, V1632, V1757  
 \if@twocolumn ..... q26, q95,  
     q152, O32, O210, O235, V95, V139,  
     V267, V278, V395, V443, V467,  
     V781, V837, V1814, V2206, V2223  
 \if@twoside ... V95, V138, V609, V667  
 \ifdt@p ..... H190, H192  
 \IfFileExists@ ..... q416, q443  
 \ifG@refundefined ..... F3, F4, F5  
 \ifh@ ..... H76, H114, H123  
 \ifin@ ... r1550, r1553, x50, x52,  
     y1, y22, y373, y475, y477, y538,  
     y551, y621, y623, y651, y703, y717,  
     y751, y763, y812, y828, y897, y900,  
     y921, y951, y954, y1017, y1020,  
     y1023, y1090, y1092, y1121, z210,  
     z214, z218, S234, S252, S483, S517  
 \ifmath@fonts ..... u204, w222  
 \ifmaybe@ic ..... C82, C91  
 \ifnot@nil ..... w343, w360, w381  
 \iftc@forced ..... D871, D882, D1150  
 \ifv@ ..... H75, H113, H122  
 \ifx ..... 266  
 \in@ ... 441, r1548, r1551, x49, x51,  
     y1, y21, y372, y474, y476, y534,  
     y547, y620, y622, y649, y701, y712,  
     y749, y760, y810, y824, y895, y898,  
     y918, y949, y952, y1014, y1018,  
     y1021, y1088, y1091, y1119, z208,  
     z212, z216, S233, S250, S480, S516  
 \in@C .....  
     v489, v490, v492, v493, y5, y6, y7, y9  
 \in@false ..... y10

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltruatex.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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

```

\in@true y12
\init@restore@glb@settings
..... w265, w268, w270
\init@restore@version
..... y62, y91, y108, y123, y124
\init@series@setup
..... v732, v737, z120, z206, z246
\input@path 10, 836,
844, 850, a109, a131, a133, a139,
a141, a147, a149, a154, a156, a166,
a233, q429, q450, q471, q498, q515
\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, V61
\install@mathalphabet
..... u587, u604, u611, y392,
y395, y481, y482, y579, y631, y634,
y641, y656, y657, y664, y1104, y1106
\is@range w376, w377
\kernel@ifnextchar
..... c87, f81, f100, f150, f678, f693, S377
\kernel@make@fragile
..... f375, f794, f795,
f796, f797, f798, f799, f800, f801,
f802, f803, f804, f805, f806, f807,
f808, f809, o24, o25, o26, o27, o28,
r172, r173, G394, G395, G396, H90,
H91, H92, H93, H166, H167, H168,
K160, K161, K162, L823, L824,
L825, L826, L827, L828, L829,
L830, L831, L832, L833, L834,
N23, N24, N25, N26, N27, R43, R44
\l@ngrel@x f74, f75, f76, f120, f167
\l@nohyphenation .. G422, G562, X269
\last@fontshape u520, u538, u555, u572
\latexrelease@postexpl g2651
\leavevmode@ifvmode o461,
o462, o470, A624, A626, A628,
A630, H115, H141, H206, H226, H227
\load@onefile@withoptions
..... 811, S861, S886, S998
\load@onefilewithoptions
..... 814, S808, S948, S1474
\lower@bound w386, w387, w398
\lst@vskip 299
\ltx@sh@ft
..... b475, r390, r397, r476, r484, r760, r767
\m@ne 269, b39
\m@th b455, b467, p13, A337,
A459, A461, A462, A465, A506,
A530, A533, A537, A540, A547,
A550, A557, A560, A642, H68, H71,
..... H106, H136, H154, H156, H172,
H191, H354, H440, H452, H479,
H490, J278, J449, K181, N214,
N237, O400, O409, O416, O437, O444
\makeph@nt H101, H103
\makesm@sh H131, H133
\mandatory@arg w414, w501, w505,
w510, w517, w519, w524, w526,
w531, w533, w546, w562, w569, w571
\math@bgroup 485, u618, w306,
w312, y53, y81, y146, y175, y221,
y243, y251, y282, C130, C131, C138
\math@egroup
..... u618, w310, w311, C131, C132, C139
\math@fonts u588, u593, w232,
w336, y60, y89, y160, y228, y258, y290
\math@fontsfalse
..... p7, r302, r329, r360, r1247, u77,
u206, u216, u239, D103, D673, D932
\math@fontstrue u204, u630
\math@version 484, u7, u336, u592,
u596, u598, u599, u601, w230, y56,
y59, y64, y65, y69, y84, y88, y93,
y94, y98, y111, y112, y113, y126,
y127, y128, y149, y151, y153, y154,
y159, y163, y165, y167, y171, y224,
y227, y231, y233, y235, y239, y254,
y257, y261, y263, y265, y269, y285,
y289, y293, y295, y297, y301, z617
\mathchar@type . y795, y805, y856,
y859, y868, y884, y989, y1000, y1063
\mathph@nt H99, H105
\mathsm@sh H129, H135
\maybe@ic C63, C64, C83
\maybe@ic@ C83
\maybe@icfalse C97
\maybe@ictrue C87
\maybe@load@fontshape
..... 352, r71, v476, v515, w127, z164
\maybe@update@bfseries@defaults .
..... z38, z257, z266, z304
\maybe@update@mdseries@defaults .
..... z39, z280, z289, z305
\mb@b J55, J66, J74, J84
\mb@l J55,
..... J59, J65, J74, J78, J83, L137, L141
\mb@r J55,
..... J59, J65, J74, J78, J83, L137, L141
\mb@t J56, J63, J75, J82
\md@def@ult z397
\mddef@ult z198, z293,
z294, z295, z334, z335, z336, z402, z443
\mddefault@previous
..... z290, z292, z331, z333, A107, A117

```

**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=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,
w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,
B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,
G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,
X=ltfinal.dtx

```

\mdseries@..... 511, 517
\mdseries@rm z111, z127, z128,
... z229, z237, z281, z293, z334, z339, z366
\mdseries@sf z112,
... z128, z238, z282, z294, z335, z340, z367
\mdseries@tt z113,
... z128, z239, z283, z295, z336, z341, z368
\meaning 253, 261, 264, 266, 267
\merge@font@series 437-
... 439, v408, v425, v426, v507, v509, w133
\merge@font@series@. v428, v433, v510
\merge@font@series@without@substitution
... 439, v460, v475, v512, w136
\merge@font@series@without@substitution@
... v460, v513
\merge@font@shape
... v626, v639, v640, v711, v716, w132
\merge@font@shape@. v642, v647, v717
\merge@font@shape@without@substitution
... v666, v719, w135
\merge@font@shape@without@substitution@
... v666, v720
\mv@<version> 485, 486
\mv@<version>@frozen 484
\mv@<version>@reset 485, 486
\n@space A625, A627, A629,
... A631, A636, A637, A638, A639, A642
\new@command
... f77, f78, f131, f165, f184, f239
\new@environment f146, f147, f159
\new@fontshape x2, x4, x22, x24
\new@mathalphabet y532, y553, y564
\new@mathgroup
... b78, b80, b98, b100, d27, u14, y412
\new@mathversion y20, y361, y368, y371
\new@module@skip c135, c147, c149
\new@moduledate c79, c98, c101, c149
\new@modulename c93, c149
\new@symbolfont y413, y445
\newcommand 251, 253, 258
\NewCommandCopy 257
\NewDocumentCommand 251
\newlinechar 266
\newmathalphabet@ x14
\newmathalphabet@@ x109
\newmathalphabet@@@ x15, x109
\nfss@catcodes 417, u19, u120, u387,
... u388, u395, u442, A40, A45, A135, V3
\nfss@text
... r315, r317, z648, C5, C122, F13
\no@alphabet@error u4, y391, y393,
... y569, y570, y584, y593, y679, y680
\noaccents@ u633, A129
\noexpand 267
\non@alpherr u612, u614, y72, y101,
... y117, y174, y242, y272, y304, y1141
\not@base
... z720, z724, z725, z726, z727,
... z728, z729, z730, z731, z732, z733, z734
\not@math@alphabet v527,
... v532, v537, v683, v687, v690, v693,
... v696, v699, v702, v705, z5, z8, z11,
... z14, z17, z20, z23, z26, z29, z255,
... z278, z308, z329, z353, z364, z381,
... z384, z406, z411, z416, z449, z454,
... z459, z475, z478, z481, z484, z487, z597
\now@and@everyjob .. d207, d213, r1011
\no@align
... b469, r390, r397, r476, r484, r760, r767
\nonline 294, h347, h730,
... l8, l15, l214, z144, z147, G177,
... G232, G249, G282, J150, S869, S987
\operator@font A643, H3,
... H4, H5, H6, H7, H8, H9, H10, H11,
... H12, H13, H14, H15, H16, H17,
... H18, H19, H20, H21, H22, H23,
... H24, H25, H26, H27, H28, H29,
... H30, H31, H32, H33, H34, H37, H40
\number@arg
... w415, w494, w496, w568, w571
\nouter@nobreak
... O181, O251, O255, O346, O364
\np@
... b311
\np@enum
... 657
\np@equation H351, H488
\np@selectfont w119
\npar
... 153
\npar@deathcycles I56, I77, I79, I80
\npatch@level 36, c1, c36, c41,
... c43, c45, c49, c58, X581, X593, X595
\npatchcmd
... 251
\nph@nt H81, H82, H83, H97
\npickup@font r181, u195, u353, u547,
... u581, w143, w170, w331, w333, w335
\npicture@ L21
\npkgcls@arg S1485, S1608
\npkgcls@candidate .. S1472, S1487,
... S1563, S1567, S1571, S1639, S1642
\npkgcls@debug S1462, S1478, S1479,
... S1480, S1481, S1482, S1539, S1540,
... S1541, S1542, S1551, S1556, S1574,
... S1583, S1598, S1632, S1633, S1634
\npkgcls@innerdate
... S1467, S1512, S1515, S1521, S1660
\npkgcls@mindate
... 828, S1492, S1501, S1517, S1522
\npkgcls@name S1484, S1527
\npkgcls@parse@date@arg .. S1486, S1497

```

**File Key:** a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltruatex.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=ltspacedtx, p=ltlogos.dtx, q=ltfiles.dtx,  
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
X=ltfinal.dtx

\pkgcls@parse@date@arg@ S1503, S1506  
 \pkgcls@parse@date@arg@version .. . . . .  
 ..... S1513, S1534  
 \pkgcls@releasedate .. . . . .  
 ..... S1472, S1568, S1572, S1643  
 \pkgcls@rollbackdate@error .. . . . .  
 ..... S1564, S1622, S1640  
 \pkgcls@show@selection .. . . . .  
 ..... S1591, S1596, S1646, S1651  
 \pkgcls@targetdate .. . . . .  
 ..... 828, S1467, S1499, S1507,  
 S1510, S1511, S1515, S1523, S1524,  
 S1537, S1545, S1560, S1562, S1592,  
 S1603, S1605, S1630, S1636, S1638  
 \pkgcls@targetlabel .. . . . .  
 ..... 828, S1467, S1500, S1520, S1535,  
 S1547, S1579, S1612, S1650, S1653  
 \pkgcls@use@this@release . S1548,  
 S1565, S1567, S1580, S1590, S1642  
 \pr@@cs .. . . . . H246, H254  
 \pr@@ct .. . . . . H249, H255  
 \pr@m@cs .. . . . . H243, H244  
 \preload@sizes .. . . . . x11, x94  
 \prepare@family@series@update ...  
 ..... 523, z117, z141,  
 z243, z407, z412, z417, z450, z455, z460  
 \pretocmd .. . . . . 250  
 \prim@s .. . . . . H240, H242, H254  
 \prime@s .. . . . . H241  
 \process@table . q40, q109, q167, y310  
 \protected .. . . . . 236  
 \protected@edef 360, 504, 974, f253,  
 s192, z546, F43, F47, F55, F59,  
 F66, J380, J397, J414, N60, O472,  
 O490, O508, X558, X567, X572, X573  
 \protected@file@percent .. . . . .  
 ..... 613, 746, G122,  
 G129, G144, G152, G153, N165, N172  
 \protected@wlog S323, S325, S339, S353  
 \protected@write .. . . . .  
 ... q201, q206, F33, N181, P14, P31  
 \protected@xdef .. . . . . f253,  
 N11, O453, O521, O537, S315, S334  
 \provide@command .. . . . . f178, f179  
 \ps@empty .. . . . . R10, X144  
 \ps@plain .. . . . . R13  
 \q@curr@file .. . . . .  
 .... S1087, S1127, S1129, S1134,  
 S1160, S1258, S1260, S1265, S1293  
 \quote@@name .. . . . . q407, q423  
 \quote@name .. . . . . q407,  
 q420, q422, q499, q501, q510, S1258  
 \r@t .. . . . . H66  
 \reenable@package@load .. . . . .  
 ..... 839, 854, 882, T444, U453  
 \reinstall@nfss@defs .. . . . . v685,  
 v726, v730, v732, v736, v737, v740  
 \relax .. . . . . 267  
 \rem@pt .. . . . . u329  
 \remove@angles .. . . . . w347, w370  
 \remove@nil .. . . . . y36  
 \remove@star .. . . . . w347, w353  
 \remove@tlig .. . . . . r1001, r1003,  
 r1005, r1030, r1035, r1071, r1073, r1075  
 \remove@to@nnil u328, w347, w373, w486  
 \renew@command .. f124, f125, f185, f193  
 \renew@environment .. . f152, f153  
 \requested@test@context .. . . . .  
 ..... 526, z503, z520, z524  
 \reserved@@b .. . . . . 500  
 \reserved@a .. . . . . 446, 526,  
 a121, a125, a126, a195, a196, a199,  
 a217, a221, a243, a250, a253, a255,  
 a256, a263, a266, a268, a269, a276,  
 a279, a281, a331, a332, a333, b193,  
 c13, c19, c34, 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, o409, o412,  
 q211, q212, q234, q243, q252, q305,  
 q360, q430, q432, q437, q439, q451,  
 q453, q458, q460, q461, q462, q472,  
 q474, q479, q481, q496, q502, q506,  
 q513, q519, q523, q552, q555, q556,  
 q558, q567, q570, q577, q580, q602,  
 q603, q604, q608, q616, q633, q634,  
 q638, q644, q666, q670, q674, r81,  
 r82, r86, r89, r97, r107, r110, r119,  
 r138, r143, r995, r999, r1049, r1058,  
 u30, u31, u32, u41, u44, u47, u63,  
 u66, u69, u105, u108, u110, u147,  
 u151, u389, u392, u519, u520, u535,  
 u538, u543, u554, u555, u568, u572,  
 u577, u604, u607, u608, u616, v435,  
 v436, v439, v440, v455, v456, v468,  
 v469, v648, v649, v652, v653, v674,  
 v675, w196, w198, w200, w210,  
 w212, w215, w344, w345, w358,  
 w359, x53, x57, y479, y488, y490,  
 y534, y537, y547, y550, y648, y650,  
 y712, y716, y760, y762, y823, y826,  
 y918, y920, y1014, y1016, y1118,

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=lpageno.dtx, F=lxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

y1120, y1136, y1138, y1139, y1144,  
z40, z41, z73, z74, z174, z175,  
z516, z517, C47, C48, C53, C54,  
C65, C68, C88, C95, G120, G121,  
G175, G176, G181, G230, G231,  
G235, G247, G248, G252, G280,  
G281, G285, G296, G297, H405,  
H406, H407, H408, H410, J57, J58,  
J61, J76, J77, J80, J145, J151,  
K241, K245, K250, K269, K360,  
K361, L199, L201, L205, L478,  
L479, L506, L507, L535, L536,  
O29, O30, O32, O33, O63, O67,  
O72, O74, O76, O78, O83, O84,  
O132, O136, O142, O145, O148,  
O151, S131, S134, S136, S228,  
S236, S240, S246, S254, S258, S385,  
S387, S388, S389, S393, S408, S410,  
S411, S412, S416, S591, S595, S601,  
S605, S683, S684, S687, S729, S733,  
S745, S746, S748, S757, S761, S773,  
S774, S776, S784, S788, S800, S801,  
S802, S804, S813, S816, S817, S819,  
S889, S938, S955, S996, S1098,  
S1099, S1101, S1229, S1230, S1232,  
S1274, S1275, S1277, S1281, S1489,  
S1494, S1546, S1547, S1578, S1579,  
S1649, S1650, S1674, S1676, T161,  
T167, T168, T169, V37, V46, V48,  
V50, V877, V897, V1997, V1999,  
V2000, V2089, V2091, V2097,  
V2100, X212, X229, X230, X231,  
X238, X239, X240, X477, X480,  
X511, X517, X518, X529, X530,  
X531, X538, X539, X540, X553,  
X554, X558, X559, X562, X563,  
X567, X568, X594, X597, X598, X615  
\reserved@b .....  
.. a122, a123, 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, o410, o411, o418, q303,  
q305, q358, q360, q497, q499, q501,  
q514, q516, q518, q602, q603, q604,  
q669, q675, r90, r97, r112, r119,  
r998, r999, r1058, r1067, r1069,  
u31, u32, u38, u95, u97, u150,  
u151, u605, u616, v454, v455, v457,  
x47, x54, x71, x73, y405, y407,  
y460, y462, y487, y488, y489, y524,  
y526, y605, y607, y652, y653, y654,  
y661, y821, y825, y827, z181, z186,  
z190, z191, z198, z199, C52, C53,  
C66, C68, C95, C96, K246, K248,  
K250, O43, O44, O112, O113,  
S229, S230, S231, S233, S248, S251,  
S385, S408, S737, S743, S746, S765,  
S771, S774, S792, S798, S802, S813,  
S820, S1143, S1144, S1147, S1148,  
S1183, S1184, S1186, S1213, S1276,  
S1277, S1280, S1281, S1316, S1317,  
S1319, S1345, S1405, S1406, S1408,  
S1435, V786, V789, V803, V806,  
V823, V826, X215, X217, X221,  
X483, X485, X489, X554, X563, X615  
\reserved@c .....  
a123, a128, f681, f684, f686, f689,  
q659, u96, u97, u606, u609, x48,  
x55, x61, x68, y33, y37, y406, y407,  
y461, y462, y525, y526, y606, y607,  
y629, y638, y653, y667, y908, y925,  
y934, y962, y973, y1031, y1044,  
y1046, z183, z186, z196, z197,  
z200, z201, C67, C69, C76, S814,  
S816, S817, S1128, S1133, S1134,  
S1152, S1160, S1166, S1188, S1196,  
S1259, S1264, S1265, S1285, S1293,  
S1299, S1321, S1328, S1376, S1377,  
S1378, S1388, S1410, S1417, S1445,  
X219, X224, X232, X477, X487,  
X508, X509, X510, X512, X513,  
X514, X515, X516, X524, X532, X617  
\reserved@d ..... a126, a129,  
e46, e51, f674, f683, q657, q659,  
x61, x68, x70, x74, y916, y925,  
y934, y970, y973, y1039, y1044,  
y1048, z188, z189, z194, z195, X618  
\reserved@e ..... o57,  
o59, o69, o71, o100, o107, o115,  
x39, x45, x70, x73, x74, y34, y39, X619  
\reserved@f ..... o58, o59,  
o70, o71, o115, r1535, r1536, r1537,  
r1538, r1540, r1547, u190, u192,  
u198, u199, w382, w393, w397,  
w401, w407, w410, w449, w486,  
w489, x27, x38, x45, x71, x73, X620  
\reset@font ..... 517, z219,  
z653, z684, z699, z714, F13, J376,  
J394, J411, O175, O371, O466,  
O485, O503, Q20, R14, V616, V675  
\restglb@settings ..... w268, w278  
\restore@mathversion .....  
..... y107, y110, y125, y133  
\restore@protect ..... f253  
\rlh@ ..... A464, A465  
\rm@def@ult ..... z397

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
X=ltfinal.dtx

\rmdef@ult 518, 526, z258, z281, z319,  
     z339, z355, z366, z398, z439, D27, D84  
 \robust@command@act .....  
     ..... 89, 91, 92, 254, 256,  
     264, f410, f411, f413, f479, f503, i89  
 \robust@command@act@chk@args ..  
     ..... 89, 90, f435, f456  
 \robust@command@act@do . 90, f421, f453  
 \robust@command@act@end ..  
     ..... 89, f418, f419, f431, f434, f454  
 \robust@command@act@loop ..  
     ..... 90, f415, f421, f451  
 \robust@command@act@loop@aux ..  
     ..... f421, f452  
 \robust@command@chk@safe ..  
     ..... 86, 95, 258,  
     f303, f414, f435, f455, f560, f572, i136  
 \s@fct@ ..... w426, w490  
 \s@fct@alias ..... w552  
 \s@fct@fixed ..... w565  
 \s@fct@gen ..... w502  
 \s@fct@genb ..... w507  
 \s@fct@sgen ..... w502  
 \s@fct@sgenb ..... w507  
 \s@fct@sub ..... w514  
 \s@fct@subf ..... w557  
 \saved@space@catcode ... X341, X410  
 \scan@fontshape ..... x7, x40, x43  
 \scan@fontshape ..... x6, x26, x37  
 \scantokens ..... 253, 264, 265  
 \scriptfont@name ..... w333, w338  
 \section ..... 252  
 \select@group ..  
     ..... 484, u589, u608, y48, y346,  
     y396, y534, y587, y596, y634, y666  
 \series@change@debug ..  
     .... z137, z144, z147, z158, z161,  
     z165, z177, z185, z190, z196, z199, z201  
 \series@check@toks .. v490, v492, v499  
 \series@drop@one@m .. v496, v500, v518  
 \series@maybe@drop@one@m ..  
     ..... u31, v483,  
     v485, v517, z168, z187, z193, z401, z402  
 \series@maybe@drop@one@m@x v486, v488  
 \seriesdefault@kernel 535, z220, z773  
 \set@mathdelimter .... y971, y1005  
 \set@color ..... J88  
 \set@curr@file .... 811, 812, 848,  
     850, 851, q225, q234, q261, q269,  
     q397, q415, S848, S1126, S1257, T267  
 \set@display@protect ..  
     ... f8, f16, f251, l7, l14, l34, l61, S326  
 \set@fontsize ..  
     ..... u317, u319, w121, w167, w177  
 \set@mathaccent ..  
     ..... y721, y729, y764, y772, y790  
 \set@mathchar ..... y845, y855  
 \set@mathdelimiter .. y922, y931, y983  
 \set@mathradical ..... y354, y1041  
 \set@mathsymbol .... y829, y837, y858  
 \set@simple@size@args ..  
     ..... w348, w361, w368, w389, w403  
 \set@size@funct@args w351, w353, w411  
 \set@size@funct@args@ ..... w411  
 \set@target@series ..  
     ..... u288, u300, v437, v441,  
     v444, v447, v470, v472, v481, v516  
 \set@typeset@protect ..  
     ..... f251, f270, K197, K235,  
     U84, U127, V603, V605, V661, V663  
 \SetMathAlphabet@ ... y541, y610, y619  
 \SetSymbolFont@ .... y431, y465, y473  
 \sf@def@ult ..... z397  
 \sf@size ..  
     p6, r302, u224, u243, u628, w328,  
     w332, D672, O409, O416, O437, O444  
 \sfdef@ult .. 518, z259, z282, z320,  
     z340, z356, z367, z399, z440, D29, D86  
 \sh@ft ..... b473  
 \show@kernel@robust@command f566, f626  
 \ShowCommand ..... 257  
 \sixt@on .. a71, b16,  
     b64, b66, b96, b97, b98, d30, u14,  
     y84, y285, y707, y709, y755, y757,  
     y816, y818, y864, y866, y904, y906,  
     y912, y914, y958, y960, y966, y968,  
     y1027, y1029, y1035, y1037, L278,  
     L293, L295, O62, O80, O131, O153,  
     V1005, V1051, V1190, V1358,  
     V1592, V1656, V1713, V1783,  
     V2043, V2052, V2108, V2124, V2157  
 \sixt@on .. 269  
 \size@update ..  
     ..... w146, w172, w185, w204, w206  
 \sizefn@info ..  
     ..... w352, w354, w362, w390, w404  
 \skip@ ..... b41, b435, b437,  
     b438, b440, o83, o442, C105, C108  
 \sp@n ..... K379  
 \split@name ..  
     . u357, u369, u483, u498, w519, w533  
 \ssf@size .. r328, r360,  
     r1246, u225, u244, u629, w328, w334  
 \string@makeletter ..  
     ... f701, S819, S857, S858, S859, T169  
 \strip@prefix .. a111, a228,  
     a323, f228, f290, f356, f698, u586

**File Key:** a=ltirch.k.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltruatex.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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

```

\strip@pt b477,
 u216, u222, u223, u224, u225,
 u238, u242, u329, u628, u629, w180
\sub@sfcnt ... w514, w515, w516, w543
\subf@sfcnt w557, w558, w559
\subst@correction u85, u91
\subst@fontshape x8, x80
\subst@size w465
\sw@slant C91, C101
\t@st@ic C90, C94
\target@meta@family@value
 z150, z175, z182, z184
\target@series@value 515,
 522, 523, z149, z157, z160, z162,
 z166, z167, z168, z191, z197, z198, z200
\tc@check@accent D109,
 D161, D163, D165, D167, D169,
 D171, D173, D175, D177, D179,
 D181, D183, D185, D187, D189,
 D191, D938, D1014, D1016, D1018
\tc@check@symbol D109,
 D211, D213, D215, D217, D219,
 D221, D223, D225, D227, D229,
 D231, D233, D235, D237, D239,
 D241, D243, D245, D247, D249,
 D251, D253, D255, D257, D259,
 D261, D263, D265, D267, D269,
 D271, D273, D275, D277, D279,
 D281, D283, D285, D287, D289,
 D291, D293, D295, D297, D299,
 D301, D303, D305, D307, D310,
 D312, D314, D316, D318, D320,
 D322, D324, D326, D328, D330,
 D332, D334, D336, D338, D340,
 D342, D344, D346, D348, D350,
 D354, D356, D358, D360, D362,
 D364, D366, D938, D1008, D1010,
 D1012, D1020, D1022, D1024,
 D1026, D1028, D1030, D1032,
 D1034, D1036, D1038, D1040,
 D1042, D1044, D1046, D1048,
 D1050, D1052, D1054, D1056,
 D1058, D1060, D1062, D1064,
 D1066, D1068, D1070, D1072,
 D1074, D1076, D1078, D1080,
 D1082, D1084, D1086, D1088,
 D1090, D1092, D1094, D1096,
 D1098, D1100, D1102, D1104,
 D1106, D1108, D1110, D1112,
 D1114, D1116, D1118, D1120, D1122
\tc@error D110, D918, D939
\tc@errorwarn
 ... D21, D76, D78, D825, D827,
 D829, D830, D877, D878, D879, D912
\tc@fake@euro D97, D352, D926, D1007
\tc@forcedfalse D871
\tc@forcedtrue D876
\tc@oldstylesubst D16, D20
\tc@subst D77, D109, D911, D938
\tc@swap@accent D112, D113
\test@font@series@context
 526, z505, z515, z535
\text@command C8, C46
\textfont@name w331, w337
\tf@size u223, u243, u627, w328, w330
\thr@@ b16, b497,
 b507, b541, w58, w254, w260, w273,
 w280, w287, w292, H363, H509,
 L287, L288, L290, L291, L329,
 L355, L388, L411, X70, X78, I232, I243
\toks@ ... 441, b41, c91, c103, c105,
 c112, c116, c119, c124, f815, f816,
 o408, o409, o414, u148, u152, u154,
 u157, u221, u226, y6, y7, y382,
 y386, y392, y395, y400, y446, y447,
 y449, y450, y480, y482, y486, y503,
 y506, y565, y577, y578, y579, y625,
 y627, y633, y641, y645, y657, y660,
 y663, y671, y673, y1094, y1096,
 y1098, y1101, y1103, y1106, y1109,
 y1141, y1142, S438, S439, S441,
 S442, V2243, V2244, V2245, V2246
\try@load@fontshape
 440, u372, u380,
 u426, u531, v479, w520, y318, y335
\try@simple@size w356, w481
\try@simples w439, w445, w449
\try@size@range ... w101, w356, w432
\try@size@substitution .. w103, w436
\tryif@simple w447, w448
\tryis@simple w448
\tt@def@ult z397
\ttdef@ult .. 518, z260, z283, z321,
 z341, z357, z368, z400, z441, D31, D88
\tw@ 269, b16
\two@digits a86, a185, a186,
 f2, w512, S1083, S1172, S1305, S1394
\type@restoreinfo w202, w207
\undeclare@... 882
\undeclare@file@substitution
 839, T247
\unqu@tefilef@und T143
\unquote@name . q401, q407, q424, T318
\unrestored@protected@xdef . f253,
 O458, O526, O542, O553, R25, R49
\update@series@target@value
 z118, z152, z173, z244

```

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

\update@uclc@with@cyrillic .... r1497, r1525, r1555, r1563  
 \upper@bound . w383, w384, w385, w398  
 \use@mathgroup ..... u595, u613, u615, w299, y63, y92, y547, y649, y652, y1119, y1143  
 \UTF@four@octets@noexpand X557, X566  
 \UTF@three@octets@noexpand ..... X556, X565  
 \UTF@two@octets@noexpand X555, X564  
 \UTFviii@four@octets X396, X401, X407  
 \UTFviii@four@octets@00 .. X396, X407  
 \UTFviii@four@octets@combine . X431  
 \UTFviii@four@octets@noexpand . X437  
 \UTFviii@four@octets@string .. X434  
 \UTFviii@invalid ..... X335, X428  
 \UTFviii@invalid@err X393, X398, X404  
 \UTFviii@invalid@err@00 .. X393, X404  
 \UTFviii@three@octets ..... X395, X400, X406  
 \UTFviii@three@octets@00 . X395, X406  
 \UTFviii@three@octets@combine . X430  
 \UTFviii@three@octets@noexpand X436  
 \UTFviii@three@octets@string .. X433  
 \UTFviii@two@octets X394, X399, X405  
 \UTFviii@two@octets@00 .. X394, X405  
 \UTFviii@two@octets@combine .. X429  
 \UTFviii@two@octets@noexpand . X435  
 \UTFviii@two@octets@string .. X432  
 \UTFviii@undefined@err ..... X392, X397, X403  
 \UTFviii@undefined@err@00 X392, X403  
 \v@false ..... H82  
 \v@true ..... H81, H83  
 \ver@{file}.{ext} ..... 811  
 \verb@balance@group . G508, G510,  
     G523, G525, G538, G540, G546, G547  
 \verb@egroup ..... G508, G511,  
     G523, G526, G538, G541, G547, G551  
 \verb@eol@error .. G548, G560, G570  
 \verbatim@font ..... G432, G453, G461, G561, G571  
 \verbatim@nolig@list ... G576, G582  
 \version@elt ..... y18, y31, y32, y379, y380,  
     y429, y449, y540, y578, y670, y1099  
 \version@list ..... y16,  
     y21, y32, y372, y380, y434, y455,  
     y474, y545, y590, y620, y675, y1112  
 \vgl@ ..... b435, b436  
 \voidb@x ..... b311, b466, t18  
 \warn@rel0i ..... x5,  
     x25, x29, x81, x85, x90, x95, x119, x140  
 \wrong@fontshape ..... u376, u513  
 \x@protect ..... f231, f242, f293, f331, f359, f530, f550  
 \xe@alloc@ ..... X42, X52  
 \xe@alloc@intercharclass ..... X21  
 \xe@ch@ck ..... X43, X47  
 \z@ ..... 269, b311  
 \z@skip ..... 299, b311  
 \zap@space ..... q261, q281, z546, Q29, S229, S390,  
     S413, S425, S590, S706, S727, S745,  
     S756, S773, S783, S800, S1096, S1227  
 tex commands:  
 \tex\_afterassignment:D ..... g2235, U49, U137  
 \tex\_aftergroup:D ... 867, U57, U143  
 \tex\_currentgrouplevel:D ..... U48, U56, U136, U142  
 \tex\_deadcycles:D ..... U91  
 \tex\_endlinechar:D ..... g1712, g1714, g1715, g1721, i342  
 \tex\_escapechar:D ..... 162, 163, g989, g999, g1168, g1177,  
     g1615, g2102, g2128, i199, T212, T226  
 \tex\_everypar:D ..... 296–298, n18, n22, n26,  
     n34, n74, n76, n86, n87, n138, n139  
 \tex\_gdef:D ..... 261, i189  
 \tex\_hskip:D ..... n60  
 \tex\_indent:D ..... n81  
 \tex\_lastnodetype:D ..... 300, n59  
 \tex\_lowercase:D ..... g1978, g2108  
 \tex\_newlinechar:D ..... i343  
 \tex\_noindent:D ..... 297, n24, n91  
 \tex\_par:D ..... 299, 300,  
     n16, n62, n69, n93, n134, n135, n136  
 \tex\_parskip:D ..... n23  
 \tex\_setbox:D ..... U50, U138  
 \tex\_shipout:D ... U125, U369, U499  
 \tex\_unkip:D ..... 300, n55  
 \tex\_vss:D ..... U332  
 \tex\_xdef:D ..... 261, i195  
 \textacutedbl ..... r915, r1145, D234, D235, D721, D971  
 \textascendercompwordmark ..... r854, D154, D685, D954  
 \textasciiaacute ..... r965, r1106, D236, D237, D722, D995  
 \textasciibreve ..... r913, r1144, D238, D239, D723, D968  
 \textasciicaron ..... r914, r1143, D240, D241, D724, D969  
 \textasciicircum ..... r286, r558, r1080  
 \textasciidieresis ..... r953, r1093, D242, D243, D725, D985

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=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,  
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspacedt, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

\textasciigrave ..... r904, r1072, D244, D245, D726, D966  
 \textasciimacron ..... r960, r1101, D246, D247, D727, D990  
 \textasciitilde ..... r287, r559, r1085  
 \textasteriskcentered ..... 373, r267, r717, r864, r865, r1207, s164, s170, D120, D577, D642, D961  
 \textbackslash ..... r268, r560, r718, r1079  
 \textbaht ..... r939, r1148, D252, D253, D731, D1101, D1102  
 \textbar ..... r269, r561, r719, r1083  
 \textbardbl ..... r270, r720, r919, r1162, s169, D127, D365, D366, D578, D650, D797, D974  
 \textbf ..... 513, C19  
 \textbigcircle ..... r729, r892, r1224, D254, D255, D732, D1053, D1054  
 \textblank ..... r861, r1221, D345, D346, D785, D1023, D1024  
 \textborn ..... r905, D256, D257, D387, D733, D1059, D1060  
 \textbraceleft ..... r271, r308, r562, r721, r1082  
 \textbraceright ..... r272, r309, r563, r722, r1084  
 \textbrokenbar ..... r951, r1091, D128, D651, D983  
 \textbullet ..... r273, r723, r921, r1171, D121, D579, D643, D976  
 \textcapitalcompwordmark ..... r853, D153, D684, D953  
 \textcelsius ..... r922, r1190, D129, D353, D354, D652, D791, D977  
 \textcent ..... r947, r1087, D130, D653, D980  
 \textcentoldstyle ..... 581, r924, D258, D259, D392, D395, D734, D1075, D1076  
 \textcircled ..... 374, 575, r279, r283, r300, r301, r730, r893, D155, D156, D655, D671, D687, D858, D1123, D1125  
 \textcircledP ..... r958, r1192, D260, D261, D735, D1117, D1118  
 \textcolonmonetary ..... r926, r1183, D313, D314, D765, D1077, D1078  
 \textcommaabove ..... r353, r355, r369, r370, r458, r459, r700, r701  
 \textcommabelow ..... r324, r326, r332, r333, r703, r704, r705, r706, r707, r708, r709, r710, r711, r712, r1244, r1447, r1448, r1449, r1450  
 \textcompsubstdefault ..... D34, D39, D91, D618, D916  
 \textcompwordmark ..... 359, r290, r291, r564, r1150  
 \textcopyleft ..... r956, D262, D263, D368, D736, D1115, D1116  
 \textcopyright ..... r283, r317, r954, r1094, D131, D654, D986  
 \textcurrency ..... r949, r1089, D335, D336, D778, D853, D857, D1011, D1012  
 \textdagger ..... r275, r312, r725, r917, r1169, s165, s171, D123, D581, D645, D972  
 \textdaggerdbl ..... r274, r313, r724, r918, r1170, s166, s172, D122, D580, D644, D973  
 \textdblyhyphen ..... r876, D266, D267, D369, D738, D1025, D1026  
 \textdblyhyphenchar ..... r912, D264, D265, D370, D737, D1071, D1072  
 \textdegree ..... r961, r1102, D132, D656, D991  
 \textdied ..... r907, D268, D269, D388, D739, D1063, D1064  
 \textdiscount ..... r941, r1182, D270, D271, D740, D1105, D1106  
 \textdiv ..... r978, r1127, D133, D657, D1005  
 \textdivorced ..... r906, r1227, D272, D273, D741, D1061, D1062  
 \textdollar ..... r255, r307, r438, r565, r798, r862, r1076, D114, D115, D623, D625, D959, D1131, D1133  
 \textdollaroldstyle ..... 581, r923, D274, D275, D393, D394, D742, D1073, D1074  
 \textdong ..... r935, r1187, D315, D316, D766, D1095, D1096  
 \textdownarrow ..... r903, r1202, D317, D318, D767, D1057, D1058  
 \texteightoldstyle ..... r886, D214, D215, D384, D710, D1043, D1044  
 \textellipsis ..... r296, r321, r1172  
 \textemdash ..... r256, r407, r411, r566, r570, r785, r1154  
 \textendash ..... r257, r408, r410, r567, r569, r786, r1153  
 \textestimated ..... r942, r1198, D329, D330, D774, D856, D1009, D1010  
 \texteuro ..... r976, r1188, D351, D352, D789, D854, D1006, D1007  
 \textexclamdown ..... r258, r412, r414, r571, r787, r1086  
 \textfiguredash ..... r410, r569, r1152, r1158  
 \textfiveoldstyle ..... r883, D216, D217, D381, D711, D1037, D1038  
 \textfloatsep ..... V728, V741, V2138, V2188, V2337  
 \textflorin ..... r925, r1141, D333, D334, D777, D978  
 \textfont ..... w337, H238

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

\textfouroldstyle ..... r882, D218,  
     D219, D380, D712, D1035, D1036  
 \textfraction ..... V1950,  
     V1953, V1978, V1981, V2130, V2331  
 \textfractionsolidus .....  
     r877, r1179, D337, D338, D780, D962  
 \textgravedbl .....  
     r916, r1146, D248, D249, D728, D970  
 \textgreater ..... r281, r572, r740, r1078  
 \textguarani ..... r929, D276,  
     D277, D391, D743, D1083, D1084  
 \textheight ..... q22, q23, q91,  
     q92, q148, q149, O257, O258, O261,  
     O287, O301, U369, V78, V225,  
     V226, V274, V399, V447, V474,  
     V645, V704, V763, V815, X142, X143  
 \texthorizontalbar r411, r570, r1155, r1160  
 \texthyphen ..... r260, r417, r574, r789  
 \texthyphenchar ..... r259, r416, r573, r788  
 \textinterrobang ..... r933, r1178,  
     D349, D350, D787, D1091, D1092  
 \textinterrobangdown ..... r934, r1228,  
     D347, D348, D786, D1093, D1094  
 \textit ..... C21  
 \textlangle ..... r888, r1219,  
     D309, D310, D762, D1047, D1048  
 \textlbrackdbl .....  
     r900, D210, D211, D389, D707, D964  
 \textleaf ..... r908, D278,  
     D279, D373, D744, D1065, D1066  
 \textleftarrow ..... r859, r1199,  
     D319, D320, D768, D1019, D1020  
 \textlegacyasteriskcentered D589, D800  
 \textlegacybardbl ..... D589, D801  
 \textlegacybullet ..... D589, D802  
 \textlegacydagger ..... D589, D804  
 \textlegacydaggerdbl ..... D589, D803  
 \textlegacyparagraph ..... D589, D805  
 \textlegacyperiodcentered .. D589, D806  
 \textlegacysection ..... D589, D807  
 \textless ..... r280, r575, r739, r1077  
 \textlira ..... r931, r1184,  
     D321, D322, D769, D1087, D1088  
 \textlnot . r957, r1099, D134, D658, D988  
 \textlquill ..... r945, r1180,  
     D280, D281, D745, D1111, D1112  
 \textmarried ..... r909, r1226,  
     D282, D283, D746, D1067, D1068  
 \textmd ..... C19  
 \textmho ..... r891, r1197,  
     D284, D285, D747, D1051, D1052  
 \textminus .....  
     r889, r1203, D343, D344, D783, D963  
 \textmu r966, r1107, D341, D342, D782, D996  
 \textmusicalnote ..... r910, r1225,  
     D286, D287, D748, D1069, D1070  
 \textnaira ..... r928, r1185,  
     D288, D289, D749, D1081, D1082  
 \textnineoldstyle ..... r887, D220,  
     D221, D385, D713, D1045, D1046  
 \textnonbreakinghyphen .....  
     r409, r568, r1151, r1156  
 \textnormal ..... C15  
 \textnumero ..... r940, r1191,  
     D331, D332, D775, D1103, D1104  
 \texttogenekcentered .... r487, r698, r699  
 \texttohm ..... r899,  
     r1196, D339, D340, D781, D855, D1008  
 \texttonehalf r974, r1116, D135, D659, D1002  
 \texttoneoldstyle ..... r879, D222,  
     D223, D377, D714, D1029, D1030  
 \texttonequarter .....  
     r973, r1115, D136, D660, D1001  
 \texttonesuperior ..... r970, r1110,  
     D137, D355, D356, D661, D792, D999  
 \textopenbullet ..... r943, r1223,  
     D290, D291, D750, D1107, D1108  
 \textordfeminine .....  
     r305, r955, r1095, D138, D662, D987  
 \textordmasculine .....  
     r306, r971, r1111, D139, D664, D1000  
 \TextOrMath s161, s164, s165, s166, s167,  
     s168, s169, s170, s171, s172, s177, s184  
 \textparagraph . r276, r310, r726, r967,  
     r1108, s168, D124, D582, D646, D997  
 \textperiodcentered ..... r277, r727,  
     r968, r1109, D125, D583, D647, D998  
 \textpertenthousand .....  
     578, r492, r937, r1174, D306, D307,  
     D308, D759, D1097, D1098, D1135  
 \textperthousand ..... r490, r920,  
     r1173, D118, D119, D639, D975, D1134  
 \textpeso ..... r930, r1189,  
     D292, D293, D751, D1085, D1086  
 \textpilcrow ..... r938, D294,  
     D295, D386, D752, D1099, D1100  
 \textpm ... r962, r1103, D140, D666, D992  
 \textquestiondown .....  
     r261, r413, r415, r576, r790, r1118  
 \textquotedbl ..... r579, r1074  
 \textquotedblleft .....  
     r262, r418, r577, r791, r1166  
 \textquotedblright .....  
     r263, r419, r578, r792, r1167  
 \textquotefontleft r264, r420, r580, r793, r1163  
 \textquotefontright r265, r421, r581, r794, r1164  
 \textquotesingle .....  
     r863, r1070, D141, D667, D960

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfsstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

\textquotestraightbase ..... r855, D142, D371, D668, D955  
 \textquotestraightdblbase ..... r856, D143, D372, D669, D956  
 \textrangle ..... r890, r1220, D311, D312, D763, D1049, D1050  
 \textrbrackdbl ..... r901, D212, D213, D390, D708, D965  
 \textrecipe ..... r932, r1193, D296, D297, D753, D1089, D1090  
 \textreferencemark ..... r969, r1177, D298, D299, D754, D1119, D1120  
 \textregistered ..... r300, r301, r959, r1100, D144, D670, D989  
 \textrightarrow ..... r860, r1201, D323, D324, D770, D1021, D1022  
 \textrm ..... C15  
 \textrquill ..... r946, r1181, D300, D301, D755, D1113, D1114  
 \textsc ..... C21  
 \textsection ..... r278, r311, r582, r728, r952, r1092, s167, D126, D584, D585, D648, D984  
 \textservicemark ..... r944, r1194, D302, D303, D756, D1109, D1110  
 \textsevenoldstyle ..... r885, D224, D225, D383, D715, D1041, D1042  
 \textsf ..... C15  
 \textsixoldstyle ..... r884, D226, D227, D382, D716, D1039, D1040  
 \textsl ..... C21  
 \textssc ..... v536, C31, C39  
 textssc ..... C25  
 \textsterling .. r266, r319, r445, r583, r805, r948, r1088, D116, D117, D624, D632, D981, D1130, D1132  
 \textstyle ..... p15, A469, H63  
 \textsubscript ..... O419  
 \textsuperscript ..... r303, r305, r306, D663, D665, D679, O402  
 \textsurd ..... r972, r1218, D304, D305, D757, D1121, D1122  
 \textsw ..... v531, C30, C38  
 textsw ..... C25  
 \TextSymbolUnavailable ..... r3, r758  
 \textthreeoldstyle ..... r881, D228, D229, D379, D717, D1033, D1034  
 \textthreequarters ..... r975, r1117, D146, D675, D1003  
 \textthreequartersemdash . r858, D145, D357, D358, D375, D674, D793, D958  
 \textthreesuperior ..... r964, r1105, D147, D359, D360, D676, D794, D994  
 \texttildelow ..... r911, r1147, D250, D251, D729, D967  
 \textttimes ..... r977, r1121, D148, D677, D1004  
 \texttrademark ..... r303, r936, r1195, D149, D678, D979  
 \textttt ..... C15  
 \texttwelveudash ..... r857, D150, D361, D362, D374, D680, D795, D957  
 \texttwooldstyle ..... r880, D230, D231, D378, D718, D1031, D1032  
 \texttwosuperior ..... r963, r1104, D151, D363, D364, D681, D796, D993  
 \textulc ..... v526, C28, C29, C35, C37  
 textulc ..... C25  
 \textunderscore .. r288, r315, r584, r1081  
 \textup ..... 443, C21  
 \textuparrow ..... r902, r1200, D325, D326, D771, D1055, D1056  
 \textvisiblespace ..... r292, r585, r1222  
 \textwidth ..... q24, q93, q150, J346, O266, V79, V144, V201, V218, V630, V640, V689, V699, V2257, V2289, X143  
 \textwon ..... r927, r1186, D327, D328, D772, D1079, D1080  
 \textyen .. r950, r1090, D152, D682, D982  
 \textzerooldstyle ..... r878, D232, D233, D376, D719, D1027, D1028  
 \TH ..... r535, r1123, X571  
 \th ..... r586, r1129, X571  
 \thanks ..... 737  
 \thanks ..... N10, N26  
 \the ..... 298  
 thebibliography (environment) ..... 776  
 \theenum ..... 657  
 \theequation .... H339, H351, H427, H488  
 \thefootnote ... O396, O521, O526, O546  
 \thempfn ..... J348, O453, O458, O537, O542, O545  
 \thempfootnote ..... J348, O398  
 \thepage ..... 878, q208, E6, F14, F34, N164, N171, N177, P15, P32, Q23, R14, V244, V275, V1827  
 \Theta ..... A299  
 \theta ..... A275  
 \thetotalpages ..... 862, 878, U348, U431  
 \thicklines ..... L124  
 \thickmuskip ..... A646, H215, H217, H230  
 \thickspace ..... H201  
 \thinlines ..... L124, L816, L833  
 \thinmuskip A644, H207, H209, H225, H231  
 \thinspace ..... o460, o466, o467, H176, H201, H238  
 \thispagestyle ..... R6

**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=ltspacedtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

```

\tilde A519
\time a179, a183
\times A396
\title 737
\title N6, N7, N21, N23, N31
tl commands:
 \c_empty_tl h66, h1114
 \c_novalue_tl 128,
 208, 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, T516, T519
 \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, U309, U314
 \tl_count:N g329, g1990, T509
 \tl_count:n g515, g520, g761, g863, g917, g1104
 \tl_gclear:N 234, 236
 \tl_gclear_new:N h127
 \tl_gput_right:Nn 225, g1018, g1223, h293, n36, y188
 \tl_gremove_once:Nn h35, h35
 \tl_gset:Nn h105, h158,
 h255, h267, h283, h321, h430, h485,
 h488, h508, h511, i123, n13, y185, T51
 \tl_gset_eq:NN h64, h290, h301
 \tl_if_blank:nTF g511, g1371, g1788, g1829, h273
 \tl_if_empty:N 239
 \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, T40, T46,
 T58, T107, T114, T116, T118, T383
 \tl_if_empty_p:N h1233, h1234, U68, U114, U367
 \tl_if_empty_p:n h542
 \tl_if_eq:NNTF g271
 \tl_if_eq:nnTF g638, g876, g2108
 \tl_if_exist:N 237
 \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 154, 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 173, g586, g2100, g2604, i236
 \tl_item:Nn g2060
 \tl_log:n h35, h37, h937
 \tl_map_function:nN g322, g512, g513, g861, g911, g2630
 \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,
 g2625, h25, h26, h27, h29, h32, h75,
 h84, h87, h96, h97, h156, h635,
 h760, h761, h762, i6, i8, i9, i10,
 n12, T8, T9, T10, T11, T59, U53, U203
 \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, T81, T82, T83, T84,
 U47, U135, U208, U234, U259, U286
 \tl_set_eq:NN g259, g265, g273, g316,
 g1321, g1404, g1477, h797, h820, i194
 \tl_show:N 162, g2083, g2090

```

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=lxref.dtx,  
G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
X=ltfinal.dtx

\tl\_show:n . . . . . [h35](#), [h36](#), [h942](#), [h1020](#)  
 \tl\_tail:N . . . . . [g1669](#)  
 \tl\_to\_str:N . . . . . [g1725](#), [g1732](#)  
 \tl\_to\_str:n . . . . . [152](#),  
   [162](#), [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](#), [T543](#)  
 \tl\_trim\_spaces:n . . . . . [262](#), [g178](#), [g349](#),  
   [g466](#), [g1310](#), [g1312](#), [g1901](#), [g1928](#),  
   [g2030](#), [g2248](#), [h251](#), [i228](#), [T41](#), [T42](#)  
 \tl\_trim\_spaces\_apply:nN . . . . .  
   [g586](#), [g2094](#), [h174](#)  
 \tl\_use:N [g1194](#), [g1241](#), [h928](#), [y195](#), [y199](#)  
 \tmspace . . . . . [634](#), [H201](#)  
 \to . . . . . [A440](#), [A442](#)  
 \today . . . . . [a184](#), [a188](#), [a196](#), [a199](#), [N33](#)  
 token commands:  
   \c\_space\_token . . . . . [g2200](#)  
   \token\_if\_active:NTF . . . . . [g1741](#)  
   \token\_if\_cs:NTF . . . . . [163](#), [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](#), [T421](#), [T433](#)  
   \token\_if\_macro:NTF . . . . . [257](#), [g2166](#), [i83](#)  
   \token\_to\_meaning:N . . . . . [i79](#), [i355](#), [i381](#)  
   \token\_to\_str:N . . . . . [144](#), [162](#), [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](#), [T241](#)  
   \toks . . . . . [298](#), [b31](#), [b63](#), [b95](#),  
     [d36](#), [n38](#), [y576](#), [y577](#), [y587](#), [y596](#), [X621](#)  
   \toksdef . . . . . [b46](#), [b63](#), [b95](#), [d222](#)  
   \tokszero . . . . . [d222](#)  
   \tolerance [b317](#), [u647](#), [u692](#), [u707](#), [R58](#), [R66](#)  
   \top . . . . . [A320](#)  
   \topfigrule . . . . . [V727](#), [V2359](#)  
   \topfraction . . . . . [O273](#), [V2325](#)  
   \topmargin . . . . . [V71](#), [V624](#), [V683](#)  
   \topmark . . . . . [V2243](#), [V2252](#)  
   \topsep . . . . . [6/6](#), [H494](#), [I2](#), [I59](#)  
   \topskip . . . . .  
     [328](#), [b388](#), [q59](#), [q127](#), [q182](#), [V128](#), [I1](#)

\totalheight . . . . . [J33](#), [J34](#), [J35](#)  
 totalpages . . . . . [862](#)  
 trace commands:  
   \trace\_stack\_levels . . . . . [X82](#)  
 \tracefloats . . . . . [V1929](#)  
 \tracefloatoff . . . . . [V1929](#)  
 \tracefloatvals . . . . . [944](#), [V1929](#)  
 \traceoff . . . . . [285](#)  
 \traceon . . . . . [285](#)  
 \tracingall . . . . . [285](#), [b491](#)  
 \tracingassigns . . . . . [b509](#), [b543](#), [b560](#), [b601](#)  
 \tracingcommands . . . . . [b343](#), [b507](#),  
   [b525](#), [b541](#), [b550](#), [b563](#), [b587](#), [b604](#)  
 \tracingfonts . . . . . [w17](#), [w54](#),  
   [w58](#), [w86](#), [w118](#), [w153](#), [w194](#), [w224](#),  
   [w238](#), [w254](#), [w260](#), [w273](#), [w280](#),  
   [w287](#), [w292](#), [w301](#), [w314](#), [w322](#), [w325](#)  
 \tracinggroups . . . . . [b499](#), [b534](#), [b572](#), [b612](#)  
 \tracingifs . . . . . [b500](#), [b535](#), [b571](#), [b611](#)  
 \tracinglostchars . . . . . [b342](#), [b492](#), [b497](#),  
   [b520](#), [b532](#), [b551](#), [b575](#), [b595](#), [b615](#)  
 \tracingmacros . . . . . [b337](#), [b506](#),  
   [b524](#), [b540](#), [b552](#), [b574](#), [b594](#), [b614](#)  
 \tracingnesting . . . . . [b502](#), [b537](#), [b569](#), [b609](#)  
 \tracingnone . . . . . [b556](#)  
 \tracingoff . . . . . [w118](#), [w322](#)  
 \tracingon . . . . . [w119](#), [w323](#)  
 \tracingonline . . . . . [b336](#), [b486](#),  
   [b562](#), [b586](#), [b603](#), [b633](#), [u656](#), [V1915](#)  
 \tracingoutput . . . . .  
   [b341](#), [b487](#), [b566](#), [b590](#), [b607](#), [b630](#)  
 \tracingpages . . . . . [b340](#), [b496](#),  
   [b519](#), [b531](#), [b551](#), [b576](#), [b596](#), [b616](#)  
 \tracingparagraphs . . . . . [b339](#), [b498](#),  
   [b521](#), [b533](#), [b552](#), [b573](#), [b593](#), [b613](#)  
 \tracingrestores . . . . . [b344](#), [b508](#),  
   [b526](#), [b542](#), [b552](#), [b561](#), [b592](#), [b602](#)  
 \tracingscantokens . . . . .  
   [b501](#), [b516](#), [b536](#), [b570](#), [b584](#), [b610](#)  
 \tracingstacklevels . . . . .  
   [963](#), [b347](#), [b350](#), [b352](#),  
   [b353](#), [b357](#), [b361](#), [b492](#), [b504](#), [b568](#)  
 \tracingstats . . . . . [b338](#), [b495](#),  
   [b518](#), [b530](#), [b550](#), [b577](#), [b597](#), [b617](#), [X2](#)  
 \triangle . . . . . [A322](#)  
 \triangleleft . . . . . [A358](#), [A482](#)  
 \triangleright . . . . . [A359](#), [A482](#)  
 \TrimSpaces . . . . . [g2626](#)  
 trivlist (environment) . . . . . [I89](#)  
 \trivlist . . . . . [653](#), [G351](#), [G401](#), [G403](#),  
   [G417](#), [G439](#), [H478](#), [K78](#), [M35](#), [M37](#), [I89](#)  
 \ttdefault . . . . . [z12](#),  
   [z216](#), [z400](#), [z417](#), [z441](#), [z460](#), [z488](#), [A50](#)

**File Key:** a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltruatex.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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

**\ttfamily** . . . . . 198, 523, z10, z11,  
z410, z458, z459, z486, z487, C17, G461  
**\ttfamily** . . . . . z420  
**\ttsubstdefault** . . . . . A20, A32, D32, D89  
**\twocolumn** . . . . . V199  
**\twocolumn[]** . . . . . 328  
**\typein** . . . . . 75, 75  
**\typein** . . . . . f31  
**\typeout** . . . . . 75  
**\typeout** . . . . . 95, 294, 440, 861, a73, a116,  
 a172, a197, a199, a211, a226, a233,  
 a244, a257, a270, a283, a321, c21,  
 c38, c43, c48, f3, f36, f43, f567, f568,  
 f601, f602, f607, l74, q200, q584,  
 q585, q591, q625, q663, q673, q676,  
 u366, v478, z138, z563, z744, z754,  
 z764, A9, A125, P8, P25, S321,  
 S337, S349, S1463, S1662, S1665,  
 U104, U115, U147, V1918, V1930,  
 X262, X575, X582, X594, X595, X603

**U**

**\u** . . . . . r237, r386,  
 r472, r589, r596, r616, r623, r754,  
 r1234, r1309, r1310, r1325, r1326,  
 r1335, r1336, r1349, r1350, r1351,  
 r1375, r1376, r1401, r1402, D163, D194  
**\uccode** . . . . . X226,  
 X234, X241, X243, X246, X248,  
 X526, X534, X541, X543, X546, X548  
**\Ucharcat** . . . . . z624  
**\uchyph** . . . . . b366  
**\ulcdefault** . . . . . v526, v605  
**\ulcshape** . . . . .  
 442, v526, v604, v698, v699, z549, C29  
**\Umathcode** . . . . .  
 . b127, d30, o476, A15, A50, A65,  
 D159, D367, G467, X146, X337, X520  
**\unboldmath** . . . . . z615  
**\UndeclareTextCommand** 574, r191, D115,  
 D117, D119, D308, D585, D1130,  
 D1131, D1132, D1133, D1134, D1135  
**\undefined** . . a9, a11, a17, a57, s71, s72,  
 s128, s129, s130, s131, s132, s133,  
 A116, A117, X154, X166, X167, X187  
**\undefinedpagestyle** . . . . . R4, R8  
**\underbar** . . . . . b456, f788, f809  
**\underbrace** . . . . . A540  
**\underline** . . . . . 661  
**\underline** . . . . . b456, J445, J446  
**\unexpanded** . . . . . 259, 261, f584, f595, C47,  
 G191, G237, G254, S740, S1491, S1493  
**\unhcopy** . . . . . b458, K345, L742, L798

**\unicodedataline** . . . . .  
 . . . . . d143, d146, d160, d161, d162  
**\UnicodeEncodingName** r981, r987, r1039,  
 r1050, r1054, r1070, r1072, r1074,  
 r1076, r1077, r1078, r1079, r1080,  
 r1081, r1082, r1083, r1084, r1085,  
 r1086, r1087, r1088, r1089, r1090,  
 r1091, r1092, r1093, r1094, r1095,  
 r1096, r1098, r1099, r1100, r1101,  
 r1102, r1103, r1104, r1105, r1106,  
 r1107, r1108, r1109, r1110, r1111,  
 r1112, r1114, r1115, r1116, r1117,  
 r1118, r1119, r1120, r1121, r1122,  
 r1123, r1124, r1125, r1126, r1127,  
 r1128, r1129, r1130, r1131, r1132,  
 r1133, r1134, r1135, r1136, r1137,  
 r1138, r1139, r1140, r1141, r1142,  
 r1143, r1144, r1145, r1146, r1147,  
 r1148, r1149, r1150, r1151, r1152,  
 r1153, r1154, r1155, r1156, r1158,  
 r1160, r1162, r1163, r1164, r1165,  
 r1166, r1167, r1168, r1169, r1170,  
 r1171, r1172, r1173, r1174, r1175,  
 r1176, r1177, r1178, r1179, r1180,  
 r1181, r1182, r1183, r1184, r1185,  
 r1186, r1187, r1188, r1189, r1190,  
 r1191, r1192, r1193, r1194, r1195,  
 r1196, r1197, r1198, r1199, r1200,  
 r1201, r1202, r1203, r1205, r1206,  
 r1207, r1218, r1219, r1220, r1221,  
 r1222, r1223, r1224, r1225, r1226,  
 r1227, r1228, r1229, r1230, r1231,  
 r1232, r1233, r1234, r1235, r1236,  
 r1237, r1238, r1239, r1240, r1241,  
 r1242, r1243, r1244, D376, D377,  
 D378, D379, D380, D381, D382,  
 D383, D384, D385, D386, D387,  
 D388, D389, D390, D391, D392, D393

**\UnicodeFontFile** . . . . . r1037  
**\UnicodeFontName** . . . . . r1038  
**\UnicodeFontTeXLigatures** . . . . r993, r1034  
**\unicoderead** . . . . .  
 . . d143, d157, d158, d159, d160, d165  
**uninstall** . . . . . d834  
**\unitlength** . . . . . 859, 860,  
 876, J53, J64, J73, J83, L5, L29,  
 L30, L32, L34, L42, L43, L44, L45,  
 L60, L63, L73, L74, L84, L85, L93,  
 L94, L107, L108, L119, L164, L176,  
 L241, L256, L316, L318, L332,  
 L340, L342, L357, L375, L377,  
 L392, L397, L399, L414, L417,  
 L422, L480, L481, L508, L509,  
 L537, L538, L612, L628, L648,

**File Key:** a= ltdirchk.dtx, b= ltpplain.dtx, c= ltvers.dtx, d= ltluatex.dtx, e= ltxexpl.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= ltspace.dtx, p= ltnologos.dtx, q= ltfilenes.dtx, r= ltoutenc.dtx, s= ltcounds.dtx, t= ltlenth.dtx, u= ltfssbas.dtx, v= ltfssaxes.dtx, w= ltfsssrc.dtx, x= ltfsscmp.dtx, y= ltfssdcl.dtx, z= ltfssini.dtx, A= fontdef.dtx, B= preload.dtx, C= ltfntcmd.dtx, D= ltttextcomp.dtx, E= ltpageno.dtx, F= ltxref.dtx, G= ltmiscen.dtx, H= ltmath.dtx, I= ltlists.dtx, J= ltnboxes.dtx, K= lttab.dtx, L= ltpictur.dtx, M= lththm.dtx, N= lsect.dtx, O= ltfloate.dtx, P= ltidxglo.dtx, Q= ltbibl.dtx, R= ltpage.dtx, S= ltcass.dtx, T= ltfiilehook.dtx, U= ltsipout.dtx, V= ltoutput.dtx, W= lthyphen.dtx, X= ltfinal.dtx

L654, L690, L691, L693, L694,  
 L697, L698, L700, L701, L712,  
 L713, L715, L716, L718, L719,  
 L721, L722, L750, L751, L753,  
 L754, L757, L758, L760, L761,  
 L772, L774, L776, L778, U326, X135  
`\unkern` ..... u675  
`\unless` ..... d151, d159, d161  
`\unlhd` ..... z732  
`\unpenalty` .....  
     ... 295, u678, u682, C116, G433, G455  
`\unrhd` ..... z734  
`\unsetattribute` ..... 42  
`\unsetattribute` ..... d82, d232  
`\unskip` ..... 295  
`\unvcopy` ..... H180  
`\Uparrow` ..... A584  
`\uparrowarrow` ..... A578  
`\upbracefill` ..... A543, A560  
`\updefault` ..... 543, v688,  
     z21, A94, A101, A103, A111, A113  
`\Updownarrow` ..... A588  
`\updownarrow` ..... A582  
`\uplus` ..... A378  
`\uppercase` ..... X558  
`\upshape` ..... 442, 444, 527, r442,  
     r732, r802, r895, v686, v687, z19,  
     z20, z549, z560, z593, z651, C24, D629  
`\Upsilon` ..... A304  
`\upsilon` ..... A286  
 use commands:  
     `\use:N` ... g711, g1238, h878, h879,  
         h919, y202, T409, T413, T415, T432  
     `\use:n` ..... 263,  
         g136, g596, g599, g759, g987, g1074,  
         g1166, g1210, g2603, g2605, g2610,  
         h226, h1068, h1194, i104, i169, i172,  
         i200, i206, i267, i300, i321, i349, i366  
     `\use:nn` ..... h849, h1084, h1362, i211  
     `\use:nnn` .....  
         171, 172, g2512, g2526, g2568, g2582  
     `\use_i:nn` ... g838, g1393, g2132, g2192  
     `\use_i:nnn` ..... 223, g2110  
     `\use_i:nnnn` ..... 263, i265  
     `\use_i_delimit_by_q_recursion_-`  
         stop:nw ..... i328  
     `\use_i_delimit_by_q_stop:nw` ...  
         ... 358, g2170  
     `\use_ii:nn` ..... 154, 223, 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` ... T216, T230, T242  
     `\use_iii:nn` ..... 223  
     `\use_iii:nnn` ..... g1794,  
         g2109, g2112, g2191, g2609, T47  
     `\use_iii:nnnn` ..... g1793  
     `\use_iv:nnnn` ..... g344, g350  
     `\use_none:n` ..... 236, g93, g600,  
         g1371, g1372, g1375, g1378, g1392,  
         g1419, g1556, g1829, g1830, g1833,  
         g1836, g2066, h7, h648, h709, h725,  
         h847, i272, i328, T131, T134, U7  
     `\use_none:nn` .....  
         ... g1003, g1550, g1788, g1789,  
         g1792, g1795, h1040, h1066, y201  
     `\use_none:nnn` ..... 154, 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` ..... J156  
`\usecounter` ..... I225, I238  
`\usefont` ..... 532, r1562,  
     u80, u282, u652, z709, D7, D613, G471  
`\UseHook` ..... 177, 178, 180,  
     181, 189, 191, 192, 203, 204, 250–  
     252, 261, 266, 835, 837, 844, h1440,  
     h1546, i212, i374, i377, q315, q321,  
     q326, w145, z263, z274, z286, z296,  
     z317, z324, z337, z344, z403, z408,  
     z413, z418, z663, z680, G173, G185,  
     G188, G212, G215, G219, G222,  
     S902, S906, S930, S934, T170,  
     T171, T174, T175, U118, U168, U371  
`\UseLegacyTextSymbols` ..... D576, D799  
`\UseOneTimeHook` .. 177, 178, 180, 181,  
     191, 192, h1440, h1547, q15, q57,  
     q70, q316, q320, q325, G14, G18,  
     G28, G29, G31, S903, S907, S929, S933  
`\usepackage` ..... 181, 182,  
     196, 835–837, 863, S611, S673, S1478  
`\UseRawInputEncoding` . X367, X423, X470  
`\UseTextAccent` . 355, r149, r150, r188,  
     D110, D111, D113, D156, D158,  
     D939, D1124, D1125, D1127, D1128  
`\UseTextSymbol` ..... 355,  
     r150, r186, D109, D352, D938, D1007  
`\usetikzlibrary` ..... 182

**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=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx,  
 r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx,  
 w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx,  
 B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx,  
 G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx,  
 M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx,  
 S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx,  
 X=ltfinal.dtx

<b>V</b>	\vline ..... K366 \vphantom ..... r497, r513, H75 \vrule ..... 954, b439, o452, r293, r295, r502, r518, w190, A558, A559, A561, A562, J165, J167, J218, J225, J444, J488, K186, K219, K347, K366, L227, L299, L302, L324, L333, L350, L359, L383, L391, L406, L413, L565, L578, L726, L782, V1851, V2260, V2293 \vskip ..... 289, 299, 309, 316 \vspace ..... o330, o400, o401, o402 \vsplit ..... V382, V429, V2242
<b>W</b>	\wedge ..... A366, A368 \whatsit ..... 41, d186 \widehat ..... A527 \widetilde ..... A526 \widowpenalties ..... b106 \widowpenalty ..... b326, u665 \width ..... J30 \wlog ..... 798, a100, b40, b145, b239, b254, b284, b299, d6, d7, d8, d54, S351, X46, X628 \wp ..... A312 \wr ..... A382 \write ..... 197, 860, 870
<b>X</b>	\x ..... c93, c96, u333, u334, X316, X318 \XeTeXcharclass ..... u640, X25, X33, X40, X53, X59, X68, X75 \XeTeXcharclassCL ..... X159 \XeTeXcharclassCM ..... X163 \XeTeXcharclassEX ..... X160 \XeTeXcharclassID ..... X157 \XeTeXcharclassIS ..... X161 \XeTeXcharclassNS ..... X162 \XeTeXcharclassOP ..... X158 \XeTeXcharglyph ..... r1035 \XeTeXdashbreakstate ..... X259 \XeTeXglyph ..... r1035 \XeTeXinterchartclasses ..... X153, X186 \XeTeXinterchartoks ..... X154, X168, X169, X170, X171, X172, X173, X174, X175, X176, X177, X178, X179, X180, X181, X182, X187, X192, X193, X194, X195, X196, X197, X198, X199, X200, X201, X202, X203, X204, X205, X206 \XeTeXmathcode ..... X147, X521 \XeTeXrevision ..... X27 \XeTeXuseglyphmetrics ..... X256, X258

**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=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssstrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfnctcmd.dtx, D=lttextcomp.dtx, E=ltpageno.dtx, F=ltxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=ltpicture.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltphphen.dtx, X=ltfinal.dtx

\XeTeXversion . . . . .	<a href="#">z623</a> , <a href="#">X27</a>	<b>Y</b>
\Xi . . . . .	<a href="#">A301</a>	\y . . . . . <a href="#">c94</a> , <a href="#">c96</a>
\xi . . . . .	<a href="#">A281</a>	\year . . . . <a href="#">a185</a> , <a href="#">c14</a> , <a href="#">S1172</a> , <a href="#">S1305</a> , <a href="#">S1394</a>
\xtxHanGlue . . . . .		
. . . . .	<a href="#">X166</a> , <a href="#">X190</a> , <a href="#">X198</a> , <a href="#">X199</a> , <a href="#">X200</a> ,	<b>Z</b>
	<a href="#">X201</a> , <a href="#">X202</a> , <a href="#">X203</a> , <a href="#">X204</a> , <a href="#">X205</a> , <a href="#">X206</a>	\Z . . . . . <a href="#">X238</a> , <a href="#">X517</a> , <a href="#">X538</a>
\xtxHanSpace . . . . .	<a href="#">X167</a> , <a href="#">X191</a> ,	\z . . . . . <a href="#">X229</a> , <a href="#">X518</a> , <a href="#">X529</a>
	<a href="#">X192</a> , <a href="#">X193</a> , <a href="#">X194</a> , <a href="#">X195</a> , <a href="#">X196</a> , <a href="#">X197</a>	\zeta . . . . . <a href="#">A273</a>

**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=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltspace.dtx, p=ltlogos.dtx, q=ltfiles.dtx, r=ltoutenc.dtx, s=ltcounts.dtx, t=ltlength.dtx, u=ltfssbas.dtx, v=ltfssaxes.dtx, w=ltfssrc.dtx, x=ltfsscmp.dtx, y=ltfssdcl.dtx, z=ltfssini.dtx, A=fontdef.dtx, B=preload.dtx, C=ltfntcmd.dtx, D=lttextcomp.dtx, E=lpageno.dtx, F=lxref.dtx, G=ltmisen.dtx, H=ltmath.dtx, I=ltlists.dtx, J=ltboxes.dtx, K=lttab.dtx, L=lpictur.dtx, M=ltthm.dtx, N=ltsect.dtx, O=ltfloat.dtx, P=ltidxglo.dtx, Q=ltbibl.dtx, R=ltpage.dtx, S=ltclass.dtx, T=ltfilehook.dtx, U=ltshipout.dtx, V=ltoutput.dtx, W=ltyphephen.dtx, X=ltfinal.dtx