

The L^AT_EX 2 _{ε} Sources

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

2023-06-01 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	3
2.2	Some bits of 2e	4
3	texsys.cfg	5
3.1	texsys.cfg	6
3.2	UNIX (web2c)	7
3.3	UNIX (other)	7
3.4	MSDOS (emtex)	7
3.5	MSDOS (other)	7
3.6	VMS (DECUS T _E X, PD VMS 3.6)	7
3.7	VMS (???)	8
3.8	MACINTOSH (OzTeX 1.6)	8
3.9	MACINTOSH (other)	8
3.10	FAKE EXAMPLE	8
4	Setting \@currdir	9
5	Setting \input@path	11

6	Filename Parsing	11
7	T_EX Versions	13
8	ltxcheck.tex	14
b	ltplain.dtx	15
1	Plain T_EX	15
c	ltvers.dtx	38
1	Version Identification	38
1.1	Declaring an all-new module	41
d	ltluatex.dtx	43
1	Overview	43
2	Core T_EX functionality	43
3	Plain T_EX interface	44
4	Lua functionality	44
4.1	Allocators in Lua	44
4.2	Lua access to T _E X register numbers	45
4.3	Module utilities	46
4.4	Callback management	46
5	Implementation	47
5.1	Minimum LuaT _E X version	47
5.2	Older L ^A T _E X/Plain T _E X setup	48
5.2.1	Fixes to <i>etex.src/etex.sty</i>	48
5.2.2	luatex specific settings	49
5.3	Attributes	50
5.4	Category code tables	50
5.5	Named Lua functions	52
5.6	Custom whatsis	52
5.7	Lua bytecode registers	53
5.8	Lua chunk registers	53
5.9	Lua loader	53
5.10	Lua module preliminaries	55
5.11	Lua module utilities	55
5.11.1	Module tracking	55
5.11.2	Module messages	56
5.12	Accessing register numbers from Lua	57
5.13	Attribute allocation	58
5.14	Custom whatsit allocation	59

5.15	Bytecode register allocation	59
5.16	Lua chunk name allocation	59
5.17	Lua function allocation	60
5.18	Lua callback management	60
5.18.1	Housekeeping	60
5.18.2	Handlers	65
5.18.3	Public functions for callback management	68
e	ltexpl.dtx	74
1	expl3-dependent code	74
1.1	Loader	74
1.2	Using expl3 code	77
2	Document-level command names for expl3 functions	78
f	ltdefns.dtx	80
1	Definitions	80
1.1	Initex initializations	80
1.2	Saved versions of T _E X primitives	80
1.3	Command definitions	81
1.4	Robust commands and protect	90
1.5	Acting on robust commands	96
1.5.1	Copying robust commands	98
1.5.2	Showing robust commands	100
1.5.3	Commands defined with \DeclareRobustCommand	101
1.5.4	Commands defined with \newcommand (with optional argument) .	103
1.5.5	Showing environments	104
1.6	Internal defining commands	106
2	Discretionary Hyphenation	110
g	ltcmd.dtx	113
1	Creating document commands	113
1.1	Variables and constants	113
1.2	Declaring commands and environments	117
1.3	Structure of xparse commands	121
1.4	Normalizing the argument specifications	125
1.5	Preparing the signature: general mechanism	134
1.6	Setting up a standard signature	134
1.7	Setting up expandable types	140
1.7.1	Copying a command and its internal structure	143
1.7.2	Showing the definition of a command	148
1.8	Grabbing arguments	153
1.9	Grabbing arguments expandably	165
1.10	Argument processors	171

1.11	Conversion to key–value form	173
1.12	Access to the argument specification	176
1.13	Utilities	178
1.14	Messages	181
1.15	User functions	187
h	lthooks.dtx	192
1	Introduction	192
2	Package writer interface	192
2.1	$\text{\LaTeX} 2_{\varepsilon}$ interfaces	192
2.1.1	Declaring hooks	192
2.1.2	Special declarations for generic hooks	193
2.1.3	Using hooks in code	194
2.1.4	Updating code for hooks	195
2.1.5	Hook names and default labels	197
2.1.6	The <code>top-level</code> label	199
2.1.7	Defining relations between hook code	200
2.1.8	Querying hooks	201
2.1.9	Displaying hook code	202
2.1.10	Debugging hook code	203
2.2	L3 programming layer (<code>expl3</code>) interfaces	203
2.3	On the order of hook code execution	206
2.4	The use of “reversed” hooks	207
2.5	Difference between “normal” and “one-time” hooks	208
2.6	Generic hooks provided by packages	209
2.7	Hooks with arguments	210
2.8	Private \LaTeX kernel hooks	212
2.9	Legacy $\text{\LaTeX} 2_{\varepsilon}$ interfaces	212
3	$\text{\LaTeX} 2_{\varepsilon}$ commands and environments augmented by hooks	213
3.1	Generic hooks	213
3.1.1	Generic hooks for all environments	214
3.1.2	Generic hooks for commands	215
3.1.3	Generic hooks provided by file loading operations	215
3.2	Hooks provided by <code>\begin{document}</code>	215
3.3	Hooks provided by <code>\end{document}</code>	216
3.4	Hooks provided by <code>\shipout</code> operations	217
3.5	Hooks provided for paragraphs	217
3.6	Hooks provided in NFSS commands	217
3.7	Hook provided by the mark mechanism	218

4	The Implementation	218
4.1	Debugging	218
4.2	Borrowing from internals of other kernel modules	219
4.3	Declarations	219
4.4	Providing new hooks	221
4.4.1	The data structures of a hook	221
4.4.2	On the existence of hooks	222
4.4.3	Setting hooks up	224
4.4.4	Disabling and providing hooks	229
4.5	Parsing a label	231
4.6	Adding or removing hook code	235
4.7	Setting rules for hooks code	255
4.8	Specifying code for next invocation	276
4.9	Using the hook	278
4.10	Querying a hook	283
4.11	Messages	287
4.12	L ^A T _E X 2 _E package interface commands	290
4.13	Deprecated that needs cleanup at some point	294
4.14	Internal commands needed elsewhere	295
i	ltcmdhooks.dtx	298
1	Introduction	298
2	Restrictions and Operational details	299
2.1	Patching	299
2.1.1	Timing	299
2.2	Commands that look ahead	300
3	Package Author Interface	300
3.1	Arguments and redefining commands	301
4	The Implementation	302
4.1	Execution plan	302
4.2	Variables	303
4.3	Variants	304
4.4	Patching or delaying	304
4.5	Patching commands	305
4.5.1	Patching by expansion and redefinition	306
4.5.2	Patching by retokenization	314
4.6	Messages	321
j	ltalloc.dtx	322
1	Counters	322
k	ltcntrl.dtx	324

1	Program control structure	324
1	lterror.dtx	328
1	Error handling and tracing	328
1.1	General commands	328
1.2	Specific errors	334
1.3	Tracing	338
m	ltpar.dtx	339
1	Paragraphs	339
1.1	Implementation	339
n	ltpara.dtx	341
1	Introduction	341
1.1	The default processing done by the engine	341
2	The new mechanism implemented for L^AT_EX	343
2.1	The provided hooks	344
2.2	Altered and newly provided commands	345
2.3	Examples	346
2.3.1	Testing the mechanism	346
2.3.2	Mark the first paragraph of each <code>itemize</code>	348
2.4	Some technical notes	348
2.4.1	Glue items between paragraphs (found with <code>fancypar</code>)	348
3	The Implementation	349
3.1	Providing hooks for paragraphs	349
3.2	The error messages	356
o	ltmeta.dtx	358
1	Introduction	358
1.1	\DocumentMetadata	358
2	The Implementation	358
p	ltspace.dtx	360

1	Spacing	360
1.1	User Commands	360
1.2	Chris' comments	360
1.3	Some immediate actions	362
1.4	The code	363
1.5	Vertical spacing	370
1.6	Horizontal space (and breaks)	375
q	ltlogos.dtx	379
1	Logos	379
r	ltfiles.dtx	380
1	File Handling	380
1.1	Safe Input Macros	393
1.2	Listing files	401
s	ltoutenc.dtx	403
1	Font encodings	403
1.1	Removing encoding-specific commands	405
1.2	The order of declarations	406
1.3	Docstrip modules	406
1.4	Definitions for the kernel	407
1.4.1	Declaration commands	407
1.4.2	Hyphenation	415
1.4.3	Miscellania	415
1.4.4	Default encodings	415
1.4.5	Math material	418
1.5	Definitions for the OT1 encoding	419
1.6	Definitions for the T1 encoding	421
1.7	Definitions for the OMS encoding	427
1.8	Definitions for the OML encoding	428
1.9	Definitions for the OT4 encoding	428
1.10	Definitions for the TS1 encoding	430
1.11	Definitions for the TU encoding	435
2	Package files	446
2.1	The fontenc package	446
t	ltcounts.dtx	449
1	Counters and Lengths	449
1.1	Environment Counter Macros	449

u	ltlength.dtx	457
1	Lengths	457
v	ltfssbas.dtx	459
1	Preliminary macros	459
2	Macros for setting up the tables	460
3	Selecting a new font	467
3.1	Macros for the user	467
3.2	Macros for loading fonts	472
4	Assigning math fonts to <i>versions</i>	480
w	ltfssaxes.dtx	487
1	Changing the font series	487
1.1	The series lookup table	487
1.2	Mapping rules for series changes	488
1.3	Changing to a new series	496
2	Changing the shape	501
2.1	Mapping rules for shape combinations	502
2.2	Changing to a new shape	504
3	Make sure we win . . .	506
x	ltfsstrc.dtx	508
1	Introduction	508
2	A driver for this document	508
3	The Implementation	509
4	Handling Options	509
5	Macros common to fam.tex and tracefnt.sty	511
5.1	General font loading	511
5.2	Math fonts setup	517
5.2.1	Outline of algorithm for math font sizes	517
5.2.2	Code for math font size setting	518
5.2.3	Other code for math	519
6	Scaled font extraction	521
6.1	Sizefunctions	529

y ltfsscmp.dtx	533
z ltfssdcl.dtx	538
1 Interface Commands	538
A ltfssini.dtx	570
1 NFSS Initialization	570
1.1 Providing math <i>versions</i>	570
2 Custom series settings for main document families	571
3 Supporting nested emphasis	588
3.1 Legacy	592
3.2 Miscellaneous	592
B fontdef.dtx	598
1 Introduction	598
2 Customization	598
3 The docstrip modules	599
4 A driver for this document	599
5 The fonttext.ltx file	599
5.1 Encodings	600
5.2 Defaults	602
6 The fontmath.ltx file	604
6.1 The font encodings used	604
6.1.1 Symbolfont and Alphabet declarations	604
6.2 Math font sizes	605
6.3 The math symbol assignments	606
6.3.1 The letters	606
6.3.2 The digits	607
6.3.3 Punctuation, brace, etc. keys	607
6.3.4 Delimitercodes for characters	607
6.4 Symbols accessed via control sequences	608
6.4.1 Greek letters	608
6.4.2 Ordinary symbols	609
6.4.3 Large Operators	609
6.4.4 Binary symbols	610
6.4.5 Relations	611
6.4.6 Arrows	612
6.4.7 Punctuation symbols	613
6.4.8 Math accents	614

6.4.9	Radicals	614
6.4.10	Over and under something, etc	614
6.4.11	Delimiters	615
6.5	Math versions of text commands	616
6.6	Other special functions and parameters	616
6.6.1	Biggggg	616
6.6.2	The log-like functions	617
6.6.3	Parameters	617
7	Default cfg files	617
C	preload.dtx	619
1	Overview	619
2	Customization	619
3	Module switches for the DOCSTRIP program	619
4	A driver for this document	620
5	The code	620
D	ltfntcmd.dtx	622
1	Introduction	622
2	The implementation	624
3	Initialization	630
E	lttextcomp.dtx	631
1	Sub-encodings	635
1.1	Unavailable in sub-encoding 1 and higher (drop symbols not working in Latin Modern)	636
1.2	Unavailable in sub-encoding 2 (majority of new OTF fonts via autoinst) and higher	636
1.3	Unavailable in sub-encoding 3 and higher	640
1.4	Unavailable in sub-encoding 4 and higher	640
1.5	Unavailable in sub-encoding 5 (most older PS fonts) and higher	640
1.6	Unavailable in sub-encoding 6 and higher	641
1.7	Unavailable in sub-encoding 7 and higher	641
1.8	Unavailable in sub-encoding 8 and higher	641
1.9	Unavailable in Sub-encoding 9 (most missing)	641
2	Unicode engine specials	642
3	Font family sub-encodings setup	642

4	Legacy symbol support for lists and footnote symbols	646
5	The <code>textcomp</code> package	651
5.1	The old <code>textcomp</code> package code	652
5.1.1	Supporting oldstyle digits	660
5.1.2	Subset encoding defaults	661
F	<code>ltpageno.dtx</code>	663
1	Page Numbering	663
G	<code>ltxref.dtx</code>	664
1	Cross Referencing	664
1.1	Cross Referencing	664
H	<code>ltmiscen.dtx</code>	671
1	Miscellaneous Environments	671
1.1	Environments	671
1.2	Center, Flushright, Flushleft	683
1.3	Verbatim	686
I	<code>ltmath.dtx</code>	693
1	Math setup	693
1.1	Math commands based on plain <code>TEX</code>	693
1.1.1	The log-like functions	693
1.1.2	Biggggg	694
1.1.3	The UNSORTED Rest	694
1.2	Math Environments	701
1.3	External options to the standard document classes	706
1.3.1	Left equation numbering	706
1.3.2	Flush left equations	706
J	<code>ltlists.dtx</code>	709
1	List, and related environments	709
1.1	List and Trivlist	710
1.2	Vertical Spacing (skips)	711
1.3	Penalties	711
1.4	Horizontal Spacing (dimens)	711
1.5	Default Values	711
1.6	Itemize and Enumerate	722

K	ltboxes.dtx	725
1	I^AT_EX Box commands	725
1.1	Some low-level constructs	740
L	lttab.dtx	742
1	Tabbing, Tabular and Array Environments	742
1.1	tabbing	742
1.2	array and tabular environments	751
M	lt pictur.dtx	767
1	Picture Mode	767
1.1	Curves	794
N	ltthm.dtx	799
1	Theorem Environments	799
O	ltsect.dtx	803
1	Sectioning Commands	803
1.1	The Title	803
1.2	Sectioning	804
1.2.1	Initializations	811
1.3	Table of Contents etc.	811
1.3.1	Convention	811
1.3.2	Commands	811
P	ltfloat.dtx	816
1	Floats	816
1.1	Floating Environments	816
1.2	Footnotes	830
Q	ltidxglo.dtx	839
1	Index and Glossary Generation	839
R	ltbibl.dtx	842
1	Bibliography Generation	842
1.1	Default definitions	846

S	ltmarks.dtx	847
1	Introduction	847
2	Design-level and code-level interfaces	848
2.1	Debugging mark code	851
3	Application examples	851
4	Legacy $\text{\LaTeX} 2\epsilon$ interface	851
4.1	Legacy design-level and document-level interfaces	851
4.2	Legacy interface extensions	852
5	Notes on the mechanism	852
6	Internal output routine functions	854
7	The Implementation	855
7.1	Allocating new mark classes	855
7.2	Updating mark structures	856
7.3	Placing and retrieving marks	860
7.4	Comparing mark values	861
7.5	Messages	862
7.6	Debugging the mark structures	863
7.7	Designer-level interfaces	864
8	$\text{\LaTeX} 2\epsilon$ integration	865
8.1	Core $\text{\LaTeX} 2\epsilon$ integration	865
8.2	Other $\text{\LaTeX} 2\epsilon$ output routines	867
T	ltpage.dtx	869
1	Page styles and related commands	869
1.1	Page Style Commands	869
1.2	How a page style makes running heads and feet	869
1.3	marking conventions	869
U	ltclass.dtx	874
1	Introduction	874
2	User interface	874
2.1	Option processing	875
3	Class and Package interface	876
3.1	Class name and version	876
3.2	Package name and version	876
3.3	Requiring other packages	876
3.4	Declaring new options	877
3.5	Safe Input Macros	878

4	Implementation	878
4.1	Hooks	905
4.2	Providing shipment	907
5	Package/class rollback mechanism	915
6	After Preamble	923
V	ltkeys.dtx	925
1	Creating and using keyval options	925
1.1	Implementation of <code>ltkeys</code>	926
1.2	Key properties	926
1.3	Main mechanism	926
1.4	The document interfaces	929
1.5	Option usage scope	930
1.6	General key setting	931
W	ltfilehook.dtx	933
1	Introduction	933
1.1	Provided hooks	933
1.2	General hooks for file reading	933
1.3	Hooks for package and class files	934
1.4	Hooks for <code>\include</code> files	935
1.5	High-level interfaces for L ^A T _E X	936
1.6	Internal interfaces for L ^A T _E X	937
1.7	A sample package for structuring the log output	937
2	The Implementation	938
2.1	Document and package-level commands	938
2.2	<code>expl3</code> helpers	939
2.3	Declaring the file-related hooks	942
2.4	Patching L ^A T _E X's <code>\InputIfFileExists</code> command	942
2.5	Declaring a file substitution	944
2.6	Selecting a file (<code>\set@curr@file</code>)	946
2.7	Replacing a file and detecting loops	949
2.7.1	The Tortoise and Hare algorithm	950
2.8	Preventing a package from loading	952
2.9	High-level interfaces for L ^A T _E X	953
2.10	Internal commands needed elsewhere	953
3	A sample package for structuring the log output	954
4	Package emulations	955
4.1	Package <code>atveryend</code> emulation	955
X	ltshipout.dtx	956

1	Introduction	956
1.1	Overloading the <code>\shipout</code> primitive	956
1.2	Provided hooks	957
1.3	Legacy L<small>A</small>T<small>E</small>X commands	959
1.4	Special commands for use inside the hooks	960
1.5	Provided L<small>A</small>T<small>E</small>X callbacks	960
1.6	Information counters	961
1.7	Debugging <code>shipout</code> code	961
2	Emulating commands from other packages	962
2.1	Emulating <code>atbegshi</code>	962
2.2	Emulating <code>everyshi</code>	963
2.3	Emulating <code>atenddvi</code>	963
2.4	Emulating <code>everypage</code>	963
3	The Implementation	964
3.1	Debugging	964
3.2	Handling the end of job hook	976
4	Legacy L<small>A</small>T<small>E</small>X 2ε interfaces	979
5	Internal commands needed elsewhere	980
6	Package emulation for compatibility	981
6.1	Package <code>atenddvi</code> emulation	981
6.2	Package <code>atbegshi</code> emulation	982
6.3	Package <code>everyshi</code> emulation	983
Y	loutput.dtx	984
1	Output Routine	984
1.1	Floats	984
1.1.1	Kludgeins	1040
1.1.2	Float control	1042
1.1.3	Float placement parameters	1055
Z	lhyphen.dtx	1058
aa	ltfinal.dtx	1060

1	Final settings	1060
1.1	Debugging	1060
1.2	Typesetting parameters	1060
1.3	Lccodes for hyphenation	1064
1.4	Hyphenation	1066
1.5	Font loading	1067
1.6	Input encoding	1068
1.7	Lccodes and uccodes	1073
1.8	Applying Patch files	1076
1.9	Freeing Memory	1077
1.10	Initialise file list	1077
1.11	Preparation for supporting PDF in backends	1078
1.12	Do some temporary work for pre-release	1078
1.13	Some last minute initializations	1078
1.14	Dumping the format	1078
Change History		1079
Index		1153

File a

ltdirchk.dtx

1 L^AT_EX System Dependent Initializations

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

The macros that must be defined are:

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

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

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

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

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

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

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

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

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

<undefined> otherwise.

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

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

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

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

```
LaTeX Error: \rubbish undefined.

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

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

2 Initialization

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

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

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

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

2.1 INITEX

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

```

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

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

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

```

2.2 Some bits of 2e

```

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

```

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

```

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

```

```

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

```

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

3 texsys.cfg

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

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

```

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

```

START

3.1 texsys.cfg

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

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

The macros that must be defined are:

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

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

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

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

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

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

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

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

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

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

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

3.2 UNIX (web2c)

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

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

3.3 UNIX (other)

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

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

3.4 MSDOS (emtex)

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

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

3.5 MSDOS (other)

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

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

3.6 VMS (DECUS T_EX, PD VMS 3.6)

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

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

3.7 VMS (???)

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

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

3.8 MACINTOSH (OzTeX 1.6)

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

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

3.9 MACINTOSH (other)

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

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

3.10 FAKE EXAMPLE

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

the following code would possibly work:

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

END

```
173 \immediate\closeout15
```

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

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

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

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

4 Setting `\@currdir`

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

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

The current date and time stamp.

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

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

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

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

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

Stop here if the file is being used unstripped.

```

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

```

5 Setting \input@path

Earlier versions of this file attempted to automatically test whether `\input@path` was required, and interactively prompt for a path if necessary. This was not found to be very reliable. The first-time installer of L^AT_EX 2_< 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_<. This will check, among other things, whether `texsys.cfg` will need to be edited and the format remade.

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

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

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

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

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

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

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

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

6 Filename Parsing

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

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

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

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

Search for the last /.

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

```

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

```

```

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

```

Remove a final dot, added earlier.

```

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

```

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

```

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

```

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

7 TeX Versions

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

```

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

```

(End of definition for \TeXversion.)

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

8 ltxcheck.tex

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

File b

ltplain.dtx

1 Plain T_EX

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

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

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

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

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

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

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

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

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

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

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

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

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

15 \catcode`@=11

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

\@ne Small constants are defined using \chardef.

\tw@ 16 \chardef\@ne=1

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

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

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

20 \chardef\@cclv=255

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

\m@ne The constant -1.

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

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

\wlog Write on log file (only)

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

(*End of 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 of 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 \newfam 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 of 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 of 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 of 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 of definition for <code>\e@alloc</code>.)</p>
<code>\e@ch@ck</code>	Extended check command. If the first range is exceeded, bump to 256 (or 266 for counts) and try again, testing the extended range.

```

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

\extrafloats 152  {/2ekernel}
              {*2ekernel | latexrelease}
              {latexrelease\IncludeInRelease{2015/10/01}
               {latexrelease           {\e@ch@ck}{Extended Allocation (checking)}%}
156  \gdef\c@ch@ck#1#2#3#4{%
157    \ifnum#1<#2\else

```

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

```

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

```

Check we are below the extended limit.

```

162    \ifnum#1<#3\relax
163    \else
164      \errmessage{No room for a new \string#4}%
165    \fi
166  \fi}%
167  {latexrelease\EndIncludeInRelease
168  {latexrelease\IncludeInRelease{2015/01/01}%
169  {latexrelease           {\e@ch@ck}{Extended Allocation (checking)}%
170  {latexrelease\gdef\c@ch@ck#1#2#3#4{%
171    {latexrelease \ifnum#1<#2\else
172    {latexrelease \ifnum#1=#2\relax
173    {latexrelease #1\c@cclvi
174    {latexrelease \ifx\c@count\c@global\advance\c@count 10 \fi
175    {latexrelease \fi
176    {latexrelease \ifnum#1<#3\relax
177    {latexrelease \else
178    {latexrelease \errmessage{No room for a new #4}%
179    {latexrelease \fi
180  {latexrelease \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\c@#1\relax
192 \ifnum\c@count>\c@z@%
193 \newinsert\c@reserved\c@a
194 \c@global\c@expandafter\c@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 \ifnum\count@<266 \ch@ck0\m@ne\insert\fi
207 \ch@ck0\count@\count
208 \ch@ck1\count@\dimen
209 \ch@ck2\count@\skip
210 \ch@ck4\count@\box
211 \global\edef\alloc@chardef\float@count\count@
212 \global\expandafter\edef\alloc@chardef
213           \csname bx@\the\float@count\endcsname\float@count
214 \@cons\@freelist{\csname bx@\the\float@count\endcsname}%
215 \expandafter\extrafloats\expandafter{\the\numexpr#1-1\expandafter}%
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 of 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@ck#1#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 of 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 \EndIncludeInRelease
289 \IncludeInRelease{0000/00/00}%
290 \newinsert}{Extended \newinsert}%
291 \let\e@insert@top\@undefined
292 \def\newinsert#1{\global\advance\insc@unt \m@ne
293 \ch@ck0\insc@unt\count
294 \ch@ck1\insc@unt\dimen
295 \ch@ck2\insc@unt\skip
296 \ch@ck4\insc@unt\box
297 \allocationnumber\insc@unt
298 \global\chardef#1\allocationnumber
299 \wlog{\string#1=\string\insert\the\allocationnumber}}
300 \EndIncludeInRelease
301 {*2ekernel}

```

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

```

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

```

(End of 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 of 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 of 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=0pt plus0pt minus0pt
314 \newbox\voidb@x % permanently void box register

```

(End of 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.

```
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

330 % \postdisplaypenalty=0
331 % \interlinepenalty=0
332 % \floatingpenalty=0, set during \insert
333 % \outputpenalty=0, set before TeX enters \output
334 \doublehyphendemerits=10000
335 \finalhyphendemerits=5000
336 \adjdemerits=10000

337 % \looseness=0, cleared by TeX after each paragraph
338 % \pausing=0
339 % \holdinginserts=0
340 % \tracingonline=0
341 % \tracingmacros=0
342 % \tracingstats=0
343 % \tracingparagraphs=0
344 % \tracingpages=0
345 % \tracingoutput=0
```

In the past L^AT_EX used the default value of 1 for \tracinglostchars because this was the best it could do. This way one would at least get a warning in the .log file. e-T_EX improved on that and supported a value of 2 to show the warning on the terminal, so we could have changed the default when we made the e-T_EX extensions required—however, we overlooked that opportunity. In 2021 this parameter was improved on again and now also accepts the value 3 (error on the terminal). This made us realize that we should change the default. Using 3 would really be the best, but for compatibility reasons we only use 2.

```
346 \tracinglostchars=2
347 % \tracingcommands=0
348 % \tracingrestores=0
```

\tracingstacklevels For LuaT_EX, the \tracingstacklevels functionality was implemented as a callback, so here we just define the count register to hold the value of the parameter.

```

349  </2ekernel>
350  <*2ekernel | latexrelease>
351  <latexrelease>\IncludeInRelease{2021/06/01}{\tracingstacklevels}%
352  <latexrelease>                                {\tracingstacklevels}%
353  \ifx\directlua\@undefined
354    % \tracingstacklevels=0 % added in 2021
355  \else
356    \newcount\tracingstacklevels
357    % Code for \tracingstacklevels defined in ltfinal.dtx
358  \fi
359  <latexrelease>\EndIncludeInRelease
360  <latexrelease>
361  <latexrelease>\IncludeInRelease{0000/00/00}{\tracingstacklevels}%
362  <latexrelease>                                {\tracingstacklevels}%
363  <latexrelease>\ifx\directlua\@undefined
364  <latexrelease>\else
365  <latexrelease>  \let\tracingstacklevels\@undefined
366  <latexrelease>\fi
367  <latexrelease>\EndIncludeInRelease
368  </2ekernel | latexrelease>
369  <*2ekernel>

(End of definition for \tracingstacklevels.)

370  \uchyph=1
371  % \lefthyphenmin=2 \righthypenmin=3 set below
372  % \globaldefs=0
373  % \maxdeadcycles=25 % INITEX does this
374  % \hangafter=1 % INITEX does this, also TeX after each paragraph
375  % \fam=0
376  % \mag=1000 % INITEX does this
377  % \escapechar='\\ % INITEX does this
378  \defaulthyphenchar='-
379  \defaultskewchar=-1
380  % \endlinechar='\^M % INITEX does this
381  % \newlinechar=-1      \LaTeX\ sets this in ltdefns.dtx.
382  \delimiterfactor=901
383  % \time=now % TeX does this at beginning of job
384  % \day=now % TeX does this at beginning of job
385  % \month=now % TeX does this at beginning of job
386  % \year=now % TeX does this at beginning of job

```

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

```

387  \showboxbreadth=-1
388  \showboxdepth=-1
389  \errorcontextlines=-1
390  \hfuzz=0.1pt
391  \vfuzz=0.1pt
392  \overfullrule=5pt
393  \maxdepth=4pt
394  \splitmaxdepth=\maxdimen
395  \boxmaxdepth=\maxdimen

```

```

396 % \lineskiplimit=0pt, changed by \normalbaselines
397 \delimitershortfall=5pt
398 \nulldelimiterspace=1.2pt
399 \scriptspace=0.5pt
400 % \mathsurround=0pt
401 % \predisplaysize=0pt, set before TeX enters $$
402 % \displaywidth=0pt, set before TeX enters $$
403 % \displayindent=0pt, set before TeX enters $$
404 \parindent=20pt
405 % \hangindent=0pt, zeroed by TeX after each paragraph
406 % \hoffset=0pt
407 % \voffset=0pt
408 %
409 % \baselineskip=0pt, changed by \normalbaselines
410 % \lineskip=0pt, changed by \normalbaselines
411 \parskip=0pt plus 1pt
412 \abovedisplayskip=12pt plus 3pt minus 9pt
413 \abovedisplayshortskip=0pt plus 3pt
414 \belowdisplayskip=12pt plus 3pt minus 9pt
415 \belowdisplayshortskip=7pt plus 3pt minus 4pt
416 % \leftskip=0pt
417 % \rightskip=0pt
418 \topskip=10pt
419 \splittopskip=10pt
420 % \tabskip=0pt
421 % \spaceskip=0pt
422 % \xspaceskip=0pt
423 \parfillskip=0pt plus 1fil

```

\normalbaselineskip We also define special registers that function like parameters:

```

424 \newskip\normalbaselineskip \normalbaselineskip=12pt
425 \newskip\normallineskip \normallineskip=1pt
426 \newdimen\normallineskiplimit \normallineskiplimit=0pt

```

(End of definition for \normalbaselineskip, \normallineskip, and \normallineskiplimit.)

\interfootlinepenalty

```

427 \newcount\interfootnotelinepenalty \interfootnotelinepenalty=100

```

(End of definition for \interfootlinepenalty.)

Definitions for preloaded fonts

\magstephalf

\magstep

```

428 \def\magstephalf{1095 }
429 \def\magstep#1{\ifcase#1 \or 1200\or 1440\or 1728\or
430 2074\or 2488\fi\relax}

```

(End of definition for \magstephalf and \magstep.)

Macros for setting ordinary text

```

\frenchspacing
\nonfrenchspacing 431 \def\frenchspacing{\sfcode`\. \sfcode`?\! \sfcode`\!\!` 
432   \sfcode`\:\!` \sfcode`\;` \sfcode`\,\!` 
433 \def\nonfrenchspacing{\sfcode`\..3000\sfcode`?3000\sfcode`\!3000%` 
434   \sfcode`\.:2000\sfcode`\;`1500\sfcode`\,,`1250 }

```

(End of definition for `\frenchspacing` and `\nonfrenchspacing`.)

`\normalbaselines`

```

435 \def\normalbaselines{\lineskip\normallineskip
436   \baselineskip\normalbaselineskip \lineskiplimit\normallineskiplimit}

```

(End of definition for `\normalbaselines`.)

`\M` Save a bit of space by using `\let` here.

`\I` 437 \def\^\M{\ } % control <return> = control <space>
438 \let\^\I\^\M % same for <tab>

(End of definition for `\M` and `\I`.)

`\lq`

`\rq` 439 \def\lq{`}
440 \def\rq{`}

(End of definition for `\lq` and `\rq`.)

`\lbrack`

`\rbrack` 441 \def\lbrack{[]}
442 \def\rbrack{[]}

(End of 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.

```

443 \def \aa {\r a}
444 \def \AA {\r A}

```

(End of definition for `\aa` and `\AA`.)

`\endgraf`

`\endline` 445 \let\endgraf=\par
446 \let\endline=\cr

(End of definition for `\endgraf` and `\endline`. These functions are documented on page 345.)

`\space`

```

447 \def\space{ }

```

(End of definition for `\space`.)

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

```

448 \let\empty\@empty

```

(End of definition for `\empty`.)

```
\null
449 \def\null{\hbox{}}

(End of definition for \null.)
```

```
\bgroup
\egroup
450 \let\bgroup=
451 \let\egroup=
```

(End of definition for \bgroup and \egroup.)

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

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

```
452 </2ekernel>
453 <*2ekernel | latexrelease>
454 <latexrelease>\IncludeInRelease{2022/06/01}{\obeylines}%
455 <latexrelease> {Add a redirection to obeylines and obeyspaces}%
```

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

```
456 \begingroup
457 \catcode`^\^M=\active % these lines must end with %
458 \gdef\obeylines{\catcode`^\^M\active%
459 \let^\^M\obeyedline%
```

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

```
460 }%
461 \global\let^\^M\par % this is in case ^M appears in a \write
462 \endgroup
```

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

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

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

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

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

```
\obeyedspace
```

```
464 \global\let\obeyedspace\space
465 \begingroup
466 \catcode`\ =\active%
467 \gdef\obeyspaces{\catcode`\ \active\let =\obeyedspace}%

```

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

```
468 \global\let =\space%
469 \endgroup
470 {/2ekernel | latexrelease}
471 <latexrelease>\EndIncludeInRelease
472 <latexrelease>\IncludeInRelease{0000/00/00}{\obeylines}%
473 <latexrelease> {Add a redirection to obeylines and obeyspaces}%
474 <latexrelease>
```

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

```
475 <latexrelease>\expandafter\let\csname obeylines \endcsname@\undefined
476 <latexrelease>\expandafter\let\csname obeyspace \endcsname@\undefined
477 <latexrelease>
478 <latexrelease>\begingroup
479 <latexrelease>\catcode`^M=\active % these lines must end with %
480 <latexrelease> \gdef\obeyspaces{\catcode`^M\active \let^M\par %
```

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

```
481 <latexrelease> }%
```

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

```
482 <latexrelease> \global\let^M\RawParEnd % this is in case ^M appears in a \write
483 <latexrelease>\endgroup
484 <latexrelease>\def\obeyspaces{\catcode`\ \active}
485 <latexrelease>
486 <latexrelease>\let\obeyedline@\undefined
487 <latexrelease>\let\obeyedspace@\undefined
488 <latexrelease>\EndIncludeInRelease
489 {*2ekernel}
```

(End of definition for `\obeyspaces` and others.)

`\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).

```
490 \long\def \loop #1\repeat{%
```

```

491 \def\iterate{\#1\relax % Extra \relax
492     \expandafter\iterate\fi
493 }
494 \iterate
495 \let\iterate\relax
496 }

```

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

```
497 \let\repeat=\fi
```

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

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

`\nointerlineskip`

`\offinterlineskip`

```

498 \def\nointerlineskip{\prevdepth-\@m\p@}
499 \def\offinterlineskip{\baselineskip-\@m\p@
500   \lineskip\z@\lineskiplimit\maxdimen}
```

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

`\vglue`

`\hglue`

```

501 \def\vglue{\afterassignment\vglue\skip@=}
502 \def\vglue{\par \dimen@\prevdepth \hrule \height\z@
503   \nobreak\vskip\skip@ \prevdepth\dimen@}
504 \def\hglue{\afterassignment\hglue\skip@=}
505 \def\hglue{\leavevmode \count@\spacefactor \vrule \width\z@
506   \nobreak\hskip\skip@ \spacefactor\count@}
```

(End of definition for `\vglue` and `\hglue`.)

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).

```
507 \def\slash{/\penalty\exhyphenpenalty}
```

(End of definition for `\slash`.)

`\break`

`\nobreak`

`\allowbreak`

```

508 \def\break{\penalty-\@M}
509 \def\nobreak{\penalty \@M}
510 \def\allowbreak{\penalty \z@}
```

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

`\filbreak`

`\goodbreak`

```

511 \def\filbreak{\par\vfil\penalty-200\vfilneg}
512 \def\goodbreak{\par\penalty-500 }
```

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

`\eject`

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

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

(End of definition for `\eject`.)

```

\removelastskip
514 \def\removelastskip{\ifdim\lastskip=\z@\else\vskip-\lastskip\fi}
(End of definition for \removelastskip.)
```

```

\smallbreak
\medbreak
\bigbreak
515 \def\smallbreak{\par\ifdim\lastskip<\smallskipamount
516   \removelastskip\penalty-50\smallskip\fi}
517 \def\medbreak{\par\ifdim\lastskip<\medskipamount
518   \removelastskip\penalty-100\medskip\fi}
519 \def\bigbreak{\par\ifdim\lastskip<\bigskipamount
520   \removelastskip\penalty-200\bigskip\fi}
```

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

```

\math
521 \def\math{\mathsurround\z@}
```

(End of definition for \math.)

\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?).

```

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

(End of definition for \underline.)

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

```

\strut
523 \newbox\strutbox
524 \def\strut{\relax\ifmmode\copy\strutbox\else\unhcopy\strutbox\fi}
```

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

\hidewidth For alignment entries that can stick out.

```

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

(End of definition for \hidewidth.)

```

\narrower
526 \def\narrower{%
527   \advance\leftskip\parindent
528   \advance\rightskip\parindent}
```

(End of definition for \narrower.)

L^AT_EX defines \ae and similar commands elsewhere.

```

529 \chardef\%='\%
530 \chardef\&='\&
531 \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

```

532 \def\leavevmode{\unhbox\voidbox}
```

(End of definition for \leavevmode.)

```

\mathhexbox
533 \def\mathhexbox#1{\mbox{$\mathchar"#1#3$}}
(End of definition for \mathhexbox.)

\ialign
534 \def\ialign{\everycr{}\tabskip\z@skip\halign} % initialized \halign
(End of definition for \ialign.)

\oalign
\o@align
535 \def\oalign#1{\leavevmode\vtop{\baselineskip\z@skip \lineskip.25ex%
536   \ialign{##\crcr#1\crcr}}}
537 \def\o@align{\lineskiplimit\z@ \oalign}
538 \def\ooalign{\lineskiplimit-\maxdimen \oalign}
(End of definition for \oalign, \o@align, and \ooalign.)

\sh@ft The definition of this macro in plain.tex was improved in about 1997; but as a result its
usage was changed and its new definition is not appropriate for LATEX.
Since the version given here has been in use by LATEX 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.
539 \def\sh@ft#1{\dimen0.00#1ex\multiply\dimen@{\fontdimen1\font
540   \kern-.0156\dimen0} % compensate for slant in lowered accents
(End of definition for \sh@ft.)

\ltx@sh@ft This is the LATEX 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).
541 \def\ltx@sh@ft #1{%
542   \dimen@ #1%
543   \kern \strip@pt
544   \fontdimen1\font \dimen@
545 } % kern by #1 times the current slant
(End of definition for \ltx@sh@ft.)
LATEX change: the text commands such as \d, \b, \c, \copyright, \TeX are now
defined elsewhere.
LATEX change: Make \t work in a moving argument. Now defined elsewhere.

\hrulefill LATEX change: \kern\z@ added to end of \hrulefill and \dotfill to make them work
\dotfill in 'tabular' and 'array' environments. (Change made 24 July 1987). LATEX change:
\leavevmode added at beginning of \dotfill and \hrulefill so that they work as
expected in vertical mode.
546 \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.
547 \def\dotfill{%
548   \leavevmode
549   \cleaders \hb@xt@ .44em{\hss.\hss}\hfill
550   \kern\z@}

```

(End of definition for \rulefill and \dotfill.)

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

551 `\sfcodes`)=0 \sfcodes`'=0 \sfcodes`\]=0`

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

Definitions related to output

`\magnification` doesn't work in LATEX.

```
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:

552 `\def\showoverfull{\tracingonline\one}`

(End of definition for `\showoverfull`.)

```
\showoutput
\loggingoutput
553 \gdef\loggingoutput{\tracingoutput\one
554   \showboxbreadth\maxdimen\showboxdepth\maxdimen\errorstopmode}
555 \gdef\showoutput{\loggingoutput\showoverfull}
556 </2ekernel>
```

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

```
\tracingall
\loggingall
557 <|latexrelease|\IncludeInRelease{2021/06/01}{\loggingall}
558 <|latexrelease|           {\tracingstacklevels and \tracinglostchars=3}%
559 <|2ekernel | latexrelease>
560 \edef\loggingall{%
561   \tracingstats\tw@
562   \tracingpages\one
563   \tracinglostchars\thr@@
564   \tracingparagraphs\one
565   \tracinggroups\one
566   \tracingifs\one
567   \tracingscantokens\one
568   \tracingnesting\one
569   \errorcontextlines\maxdimen
570   \ifdefined\tracingstacklevels \tracingstacklevels\maxdimen \fi
571   \noexpand \loggingoutput
572   \tracingmacros\tw@
573   \tracingcommands\thr@@
574   \tracingrestores\one
575   \tracingassigns\one
576 }%
577 \def\tracingall{\showoverfull\loggingall}
578 </2ekernel | latexrelease>
579 <|latexrelease|\EndIncludeInRelease
580 <|latexrelease|
581 <|latexrelease|\IncludeInRelease{2015/01/01}{\loggingall}{etex tracing}%
582 <|latexrelease|\ifx\tracingscantokens\undefined
583 <|latexrelease|\gdef\loggingall{%
```

```

584 〈latexrelease〉 \tracingstats\tw@  

585 〈latexrelease〉 \tracingpages\@ne  

586 〈latexrelease〉 \tracinglostchars\@ne  

587 〈latexrelease〉 \tracingparagraphs\@ne  

588 〈latexrelease〉 \errorcontextlines\maxdimen  

589 〈latexrelease〉 \loggingoutput  

590 〈latexrelease〉 \tracingmacros\tw@  

591 〈latexrelease〉 \tracingcommands\tw@  

592 〈latexrelease〉 \tracingrestores\@ne  

593 〈latexrelease〉 }%  

594 〈latexrelease〉\else  

595 〈latexrelease〉\gdef\loggingall{%
596 〈latexrelease〉 \tracingstats\tw@  

597 〈latexrelease〉 \tracingpages\@ne  

598 〈latexrelease〉 \tracinglostchars\tw@  

599 〈latexrelease〉 \tracingparagraphs\@ne  

600 〈latexrelease〉 \tracinggroups\@ne  

601 〈latexrelease〉 \tracingifs\@ne  

602 〈latexrelease〉 \tracingscantokens\@ne  

603 〈latexrelease〉 \tracingnesting\@ne  

604 〈latexrelease〉 \errorcontextlines\maxdimen  

605 〈latexrelease〉 \loggingoutput  

606 〈latexrelease〉 \tracingmacros\tw@  

607 〈latexrelease〉 \tracingcommands\thr@@  

608 〈latexrelease〉 \tracingrestores\@ne  

609 〈latexrelease〉 \tracingassigns\@ne  

610 〈latexrelease〉}%
611 〈latexrelease〉\fi  

612 〈latexrelease〉\gdef\tracingall{\showoverfull\loggingall}  

613 〈latexrelease〉\EndIncludeInRelease  

614 〈latexrelease〉  

615 〈latexrelease〉\IncludeInRelease{0000/00/00}{\loggingall}{etex tracing}%
616 〈latexrelease〉\gdef\loggingall{\tracingcommands\tw@\tracingstats\tw@  

617 〈latexrelease〉 \tracingpages\@ne\tracinglostchars\@ne  

618 〈latexrelease〉 \tracingmacros\tw@\tracingparagraphs\@ne\tracingrestores\@ne  

619 〈latexrelease〉 \errorcontextlines\maxdimen\loggingoutput}  

620 〈latexrelease〉 \gdef\tracingall{\loggingall\showoverfull}  

621 〈latexrelease〉\EndIncludeInRelease

```

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

```
\tracingnone
622 〈latexrelease〉\IncludeInRelease{2015/01/01}{\tracingnone}%
623 〈latexrelease〉                                         {turn off etex tracing}%
624 〈*2ekernel | latexrelease〉
625 \edef\tracingnone{%
626   \tracingassigns\z@  

627   \tracingrestores\z@  

628   \tracingonline\z@  

629   \tracingcommands\z@  

630   \showboxdepth\m@ne  

631   \showboxbreadth\m@ne  

632   \tracingoutput\z@  

633   \errorcontextlines\m@ne
}
```

```

634 \ifdefinable\tracingstacklevels {\tracingstacklevels\z@ \fi
635 \tracingnesting\z@
636 \tracingscantokens\z@
637 \tracingifs\z@
638 \tracinggroups\z@
639 \tracingparagraphs\z@
640 \tracingmacros\z@
641 \tracinglostchars\@ne
642 \tracingpages\z@
643 \tracingstats\z@
644 }%
645 </2ekernel | latexrelease>
646 <latexrelease>\EndIncludeInRelease
647 <latexrelease>
648 <latexrelease>\IncludeInRelease{2015/01/01}{\tracingnone}%
649 <latexrelease>                                {turn off etex tracing}%
650 <latexrelease>\ifx\tracingscantokens\@undefined
651 <latexrelease>\def\tracingnone{%
652 <latexrelease>  \tracingonline\z@
653 <latexrelease>  \tracingcommands\z@
654 <latexrelease>  \showboxdepth\m@ne
655 <latexrelease>  \showboxbreadth\m@ne
656 <latexrelease>  \tracingoutput\z@
657 <latexrelease>  \errorcontextlines\m@ne
658 <latexrelease>  \tracingrestores\z@
659 <latexrelease>  \tracingparagraphs\z@
660 <latexrelease>  \tracingmacros\z@
661 <latexrelease>  \tracinglostchars\@ne
662 <latexrelease>  \tracingpages\z@
663 <latexrelease>  \tracingstats\z@
664 <latexrelease>}%
665 <latexrelease>\else
666 <latexrelease>\def\tracingnone{%
667 <latexrelease>  \tracingassigns\z@
668 <latexrelease>  \tracingrestores\z@
669 <latexrelease>  \tracingonline\z@
670 <latexrelease>  \tracingcommands\z@
671 <latexrelease>  \showboxdepth\m@ne
672 <latexrelease>  \showboxbreadth\m@ne
673 <latexrelease>  \tracingoutput\z@
674 <latexrelease>  \errorcontextlines\m@ne
675 <latexrelease>  \tracingnesting\z@
676 <latexrelease>  \tracingscantokens\z@
677 <latexrelease>  \tracingifs\z@
678 <latexrelease>  \tracinggroups\z@
679 <latexrelease>  \tracingparagraphs\z@
680 <latexrelease>  \tracingmacros\z@
681 <latexrelease>  \tracinglostchars\@ne
682 <latexrelease>  \tracingpages\z@
683 <latexrelease>  \tracingstats\z@
684 <latexrelease>}%
685 <latexrelease>\fi
686 <latexrelease>\EndIncludeInRelease
687 <latexrelease>

```

```

688 〈\latexrelease〉\IncludeInRelease{0000/00/00}{\tracingnone}%
689 〈\latexrelease〉                                     {turn off etex tracing}%
690 〈\latexrelease〉\let\tracingnone\@undefined
691 〈\latexrelease〉\EndIncludeInRelease

```

(*End of definition for \tracingnone.*)

\hideoutput

```

692 〈*2ekernel | \latexrelease〉
693 〈\latexrelease〉\IncludeInRelease{2015/01/01}{\hideoutput}%
694 〈\latexrelease〉                                     {hide output from tracing}%
695 \def\hideoutput{%
696   \tracingoutput\z@%
697   \showboxbreadth\m@ne%
698   \showboxdepth\m@ne%
699   \tracingonline\m@ne%
700 }%
701 〈\latexrelease〉\EndIncludeInRelease
702 〈\latexrelease〉
703 〈\latexrelease〉\IncludeInRelease{0000/00/00}{\hideoutput}%
704 〈\latexrelease〉                                     {hide output from tracing}%
705 〈\latexrelease〉\let\hideoutput\@undefined
706 〈\latexrelease〉\EndIncludeInRelease
707 〈/2ekernel | \latexrelease〉

```

(*End of definition for \hideoutput.*)

L^AT_EX change: \showhyphens Defined later.

Punctuation affects the spacing.

```

708 〈*2ekernel〉
709 \nonfrenchspacing
710 〈/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` A \patchlevel 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 date 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 `\patchlevel` 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 <!*2ekernel | latexrelease>\edef\latexreleaseversion
6 <!*2ekernel | latexrelease>
7 {2023-06-01}
8 </!2ekernel | latexrelease>
9 <!*2ekernel>
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 `\patchlevel` 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 \edef\development@branch@name{\}
```

(End of 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\count@>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 !!!!!!! ^^J%
29 }
30   \errhelp{To avoid this error message, obtain new LaTeX sources.}
31   \errmessage{LaTeX source files more than 5 years old!}
32 \fi
33 \let\reserved@a\relax
34 \fi

```

We store release info in the toks \LaTeXReleaseInfo to be used in \everyjob but also when \end{document} is executed. Instead of using \typeout we use \show@release@info so that we can write to the log only by changing that to \wlog.

```

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

```

```

68      \fi
69  \fi
70 </2ekernel>

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

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

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

113 \def\@IncludeInRelease#1[#2]{\@IncludeInRelease@se{#2}}
114 \def\@IncludeInRelease@se#1#2#3{%
115   \toks@{[#1] #3}%
116   \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 ...

```

117   \ifnum\expandafter\@parse@version#1//00@nil
118     >\expandafter\@parse@version\fmtversion//00@nil
119   \GenericInfo{}{Skipping: \the\toks@}%
120   \expandafter\expandafter\expandafter\@gobble@IncludeInRelease
121 \else
122   \GenericInfo{}{Applying: \the\toks@}%
123   \@includeinreleasetrue
124   \expandafter\let\csname\string#2+\currname+IIR\endcsname\empty
125   \fi
126 \else
127   \GenericInfo{}{Already applied: \the\toks@}%
128   \expandafter\@gobble@IncludeInRelease
129 \fi
130 }

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

140 \long\def\@gobble@IncludeInRelease#1\EndIncludeInRelease{%
141   \@includeinreleasefalse
142   \@check@IncludeInRelease#1\IncludeInRelease\@check@IncludeInRelease
143   \@end@check@IncludeInRelease}

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

```

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

1.1 Declaring an all-new module

<pre>\if@skipping@module \NewModuleRelease \EndModuleRelease \new@module@skip \new@modulename \new@moduledate</pre>	<p>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.</p>
---	---

```

152 \let\if@skipping@module\iffalse
153 \def\@skipping@moduletrue{\let\if@skipping@module\iftrue}
154 \def\@skipping@modulefalse{\let\if@skipping@module\iffalse}
```

```

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

```

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

202 `</2ekernel | latexrelease>`

File d

ltluatex.dtx

1 Overview

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

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

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

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

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

2 Core TeX functionality

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

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

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

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

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

```

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

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

```

3 Plain T_EX interface

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

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

4 Lua functionality

4.1 Allocators in Lua

```

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

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

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

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

new_luafunction luatexbase.new_luafunction(\functionname)

```

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

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

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

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

4.2 Lua access to TeX register numbers

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

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

As an example, consider the input:

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

\test{undefinedrubbish}

\test{space}

\test{hbox}

\test{@MM}

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

\test{strutbox}

\test{sixt@@n}

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

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

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

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

4.3 Module utilities

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

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

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

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

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

4.4 Callback management

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

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

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

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

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

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

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

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

declare_callback_rule `luatexbase.declare_callback_rule(\langle name \rangle, \langle first \rangle, \langle relation \rangle, \langle second \rangle)` Adds an ordering constraint between two callback functions for callback $\langle name \rangle$.
The kind of constraint added depends on $\langle relation \rangle$:

before The callback function with description $\langle first \rangle$ will be executed before the function with description $\langle second \rangle$.

after The callback function with description $\langle first \rangle$ will be executed after the function with description $\langle second \rangle$.

incompatible-warning When both a callback function with description $\langle first \rangle$ and with description $\langle second \rangle$ is registered, then a warning is printed when the callback is executed.

incompatible-error When both a callback function with description $\langle first \rangle$ and with description $\langle second \rangle$ is registered, then an error is printed when the callback is executed.

unrelated Any previously declared callback rule between $\langle first \rangle$ and $\langle second \rangle$ gets disabled.

Every call to `declare_callback_rule` with a specific callback $\langle name \rangle$ and descriptions $\langle first \rangle$ and $\langle second \rangle$ overwrites all previous calls with same callback and descriptions.

The callback functions do not have to be registered yet when the functions is called. Only the constraints for which both callback descriptions refer to callbacks registered at the time the callback is called will have an effect.

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\lualatexversion<60 %
4    \wlog{*****}
5    \wlog{* LuaTeX version too old for ltluatex support *}
6    \wlog{*****}
7    \expandafter\endinput
8
9 \fi
10

```

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\etalloc\undefined
16   In pre-2014 LATEX, or plain TEX, load etex.{sty,src}.
17   \ifx\documentclass\undefined
18     \ifx\loccount\undefined
19       \input{etex.src}%
20     \fi
21     \catcode`\@=11 %
22     \outer\expandafter\def\csname newfam\endcsname
23       {\alloc@8\fam\chardef\et@xmaxfam}
24   \else
25     \RequirePackage{etex}
26     \expandafter\def\csname newfam\endcsname
27       {\alloc@8\fam\chardef\et@xmaxfam}
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 \count270=\et@xmaxregs % locally allocates \count registers
32 \count271=\et@xmaxregs % ditto for \dimen registers
33 \count272=\et@xmaxregs % ditto for \skip registers
34 \count273=\et@xmaxregs % ditto for \muskip registers
35 \count274=\et@xmaxregs % ditto for \box registers
36 \count275=\et@xmaxregs % ditto for \toks registers
37 \count276=\et@xmaxregs % ditto for \marks classes

```

and 256 or 16 fam. (Done above due to plain/L^AT_EX 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\c@alloc@top=65535
48 \let\c@alloc\chardef\chardef
49 \def\c@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\c@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}%
66 }
```

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

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

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

```
72 \fi
73 
```

5.3 Attributes

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

```

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

```

(End of definition for `\newattribute`.)

- `\setAttribute` Handy utilities.

```

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

```

(End of definition for `\setAttribute` and `\unsetattribute`.)

5.4 Category code tables

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

```

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

```

(End of 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\setrangepage#1#2#3{%
97   \ifnum#1>#2 %
98     \expandafter\@gobble
99   \else
100     \expandafter\@firstofone
101   \fi
102   {%
103     \catcode#1=#3 %
104     \expandafter\setrangepage\expandafter
105     {\number\numexpr#1 + 1\relax}{#2}{#3}
106   }%

```

```

107   }
108   \catcodetable\catcodetable@initex
109     \catcode0=12 %
110     \catcode13=12 %
111     \catcode37=12 %
112     \setrangingcatcode{65}{90}{12}%
113     \setrangingcatcode{97}{122}{12}%
114     \catcode92=12 %
115     \catcode127=12 %
116     \savecatcodetable\catcodetable@string
117   \endgroup
118 }
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 of 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 of definition for \newluafunction.)

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

```

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

```

(End of definition for \newluacmd and \newprotectedluacmd.)

5.6 Custom whatsits

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

```

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

```

(End of definition for \newwhatsit.)

5.7 Lua bytecode registers

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

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

(End of definition for \newluabytecode.)

5.8 Lua chunk registers

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

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

(End of definition for \newluachunkname.)

5.9 Lua loader

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

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

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

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

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

```

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

235 \textrun{\IncludeInRelease{0000/00/00}}
236 \textrun{\newluafunction{LuaTeX}%
237 \let\alloc@attribute@count\undefined
238 \let\newattribute\undefined
239 \let\setattribute\undefined
240 \let\unsetattribute\undefined
241 \let\alloc@ccodetable@count\undefined
242 \let\newcatcodetable\undefined
243 \let\catcodetable@initex\undefined
244 \let\catcodetable@string\undefined
245 \let\catcodetable@latex\undefined
246 \let\catcodetable@atletter\undefined
247 \let\alloc@luafunction@count\undefined
248 \let\newluafunction\undefined
249 \let\alloc@luafunction@count\undefined
250 \let\newwhatsit\undefined
251 \let\alloc@whatsit@count\undefined
252 \let\newluabytecode\undefined
253 \let\alloc@bytecode@count\undefined
254 \let\newluachunkname\undefined
255 \let\alloc@luachunk@count\undefined
256 \directlua{luatexbase.uninstall()}
257 \textrun{\EndIncludeInRelease}

```

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

```

258 \textrun{\IncludeInRelease{2017/01/01}%
259 \textrun{\fontencoding{TU in everyjob}%
260 \textrun{\fontencoding{TU}\let\encodingdefault\f@encoding
261 \textrun{\ifx\directlua\undefined\else
262 {2ekernel}\everyjob\expandafter{%
263 {2ekernel} \the\everyjob
264 {*2ekernel,luatexrelease}
265 \directluat%
266 if xpcall(function ()%
267 require('luaotfloat-main')%
268 end,texio.write_nl) then %
269 local _void = luaotfloat.main ()%
270 else %
271 texio.write_nl('Error in luaotfloat: reverting to OT1')%
272 tex.print('\string\\def\string\\encodingdefault{OT1}')%
273 end %
274 }%

```

```

275  \let\f@encoding\encodingdefault
276  \expandafter\let\csname ver@luaotfload.sty\endcsname\fmtversion
277  </2ekernel, latexrelease>
278  <latexrelease>\fi
279  <2ekernel> }
280  <latexrelease>\EndIncludeInRelease
281  <latexrelease>\IncludeInRelease{0000/00/00}%
282  <latexrelease> {\fontencoding}{TU in everyjob}%
283  <latexrelease>\fontencoding{OT1}\let\encodingdefault\f@encoding
284  <latexrelease>\EndIncludeInRelease
285  <2ekernel | latexrelease>\fi
286  </2ekernel | tex | latexrelease>

```

5.10 Lua module preliminaries

287 `(*lua)`

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

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

```

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

```

(*End of definition for luatexbase.*)

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

```

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

295 local luatexbase_warning
296 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.

```
297 local modules = modules or { }
```

(*End of definition for modules.*)

provides_module Local function to write to the log.

```

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

```

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

```

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

```

(End of 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 T_EX. For errors we have to make some changes. Here we give the text of the error in the L^AT_EX format then force an error from Lua to halt the run. Splitting the message text is done using \n which takes the place of \MessageBreak.

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

```

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

```

```

340         .. tex.inputlineno, "\n", "\n" .. cont .. " "
341     )
342     .. "\n"
343 end

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

(End of definition for module_info, module_warning, and module_error.)
Dedicated versions for the rest of the code here.

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

```

5.12 Accessing register numbers from Lua

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

```

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

```

```

383     luaregisterbasetable[hashtokens[i][1]] =
384         hashtokens[i][2]
385     else
386         luaregisterbasetable[j] = createtoken(i).mode
387     end
388 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.

```

389 local registernumber
390 if luatexversion < 80 then
391     function registernumber(name)
392         local nt = hashtokens[name]
393         if(nt and luaregisterbasetable[nt[1]]) then
394             return nt[2] - luaregisterbasetable[nt[1]]
395         else
396             return false
397         end
398     end
399 else
400     function registernumber(name)
401         local nt = createtoken(name)
402         if(luaregisterbasetable[nt.cmdname]) then
403             return nt.mode - luaregisterbasetable[nt.cmdname]
404         else
405             return false
406         end
407     end
408 end
409 luatexbase.registernumber = registernumber

```

(End of 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.

```

410 local attributes=setmetatable(
411 {}, {
412     __index = function(t,key)
413         return registernumber(key) or nil
414     end}
415 )
416 )
417 luatexbase.attributes = attributes
418 local attribute_count_name =
419             attribute_count_name or "e@alloc@attribute@count"
420 local function new_attribute(name)
421     tex_setcount("global", attribute_count_name,
422                 tex_count[attribute_count_name] + 1)
423     if tex_count[attribute_count_name] > 65534 then
424         luatexbase_error("No room for a new \\attribute")

```

```

425   end
426   attributes[name] = tex_count[attribute_count_name]
427   luatexbase_log("Lua-only attribute " .. name .. " = " ..
428                   tex_count[attribute_count_name])
429   return tex_count[attribute_count_name]
430 end
431 luatexbase.new_attribute = new_attribute

```

(End of definition for `new_attribute`.)

5.14 Custom whatsit allocation

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

```

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

```

(End of 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.

```

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

```

(End of 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.

```

457 local chunkname_count_name =
458           chunkname_count_name or "e@alloc@luachunk@count"
459 local function new_chunkname(name)

```

```

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

```

(End of definition for `new_chunkname`.)

5.17 Lua function allocation

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

```

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

```

(End of 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.

Actually there are two tables: `realcallbacklist` directly contains the entries as described above while `callbacklist` only directly contains the already sorted entries. Other entries can be queried through `callbacklist` too which triggers a resort.

Additionally `callbackrules` describes the ordering constraints: It contains two element tables with the descriptions of the constrained callback implementations. It can

additionally contain a `type` entry indicating the kind of rule. A missing value indicates a normal ordering constraint.

```

486 local realcallbacklist = {}
487 local callbackrules = {}
488 local callbacklist = setmetatable({}, {
489   __index = function(t, name)
490     local list = realcallbacklist[name]
491     local rules = callbackrules[name]
492     if list and rules then
493       local meta = {}
494       for i, entry in ipairs(list) do
495         local t = {value = entry, count = 0, pos = i}
496         meta[entry.description], list[i] = t, t
497       end
498       local count = #list
499       local pos = count
500       for i, rule in ipairs(rules) do
501         local rule = rules[i]
502         local pre, post = meta[rule[1]], meta[rule[2]]
503         if pre and post then
504           if rule.type then
505             if not rule.hidden then
506               assert(rule.type == 'incompatible-warning' and luatexbase_warning
507                 or rule.type == 'incompatible-error' and luatexbase_error)(
508                 "Incompatible functions \".. rule[1] .. \" and \".. rule[2]
509                 .. \" specified for callback \".. name .. \".")
510             rule.hidden = true
511           end
512         else
513           local post_count = post.count
514           post.count = post_count+1
515           if post_count == 0 then
516             local post_pos = post.pos
517             if post_pos ~= pos then
518               local new_post_pos = list[pos]
519               new_post_pos.pos = post_pos
520               list[post_pos] = new_post_pos
521             end
522             list[pos] = nil
523             pos = pos - 1
524           end
525           pre[#pre+1] = post
526         end
527       end
528     end
529     for i=1, count do -- The actual sort begins
530       local current = list[i]
531       if current then
532         meta[current.value.description] = nil
533         for j, cur in ipairs(current) do
534           local count = cur.count
535           if count == 1 then
536             pos = pos + 1
537             list[pos] = cur

```

```

538         else
539             cur.count = count - 1
540         end
541     end
542     list[i] = current.value
543   else
544     -- Cycle occurred. TODO: Show cycle for debugging
545     -- list[i] = ...
546     local remaining = {}
547     for name, entry in next, meta do
548       local value = entry.value
549       list[#list + 1] = entry.value
550       remaining[#remaining + 1] = name
551     end
552     table.sort(remaining)
553     local first_name = remaining[1]
554     for j, name in ipairs(remaining) do
555       local entry = meta[name]
556       list[i + j - 1] = entry.value
557       for _, post_entry in ipairs(entry) do
558         local post_name = post_entry.value.description
559         if not remaining[post_name] then
560           remaining[post_name] = name
561         end
562       end
563     end
564     local cycle = {first_name}
565     local index = 1
566     local last_name = first_name
567     repeat
568       cycle[last_name] = index
569       last_name = remaining[last_name]
570       index = index + 1
571       cycle[index] = last_name
572     until cycle[last_name]
573     local length = index - cycle[last_name] + 1
574     table.move(cycle, cycle[last_name], index, 1)
575     for i=2, length//2 do
576       cycle[i], cycle[length + 1 - i] = cycle[length + 1 - i], cycle[i]
577     end
578     error('Cycle occurred at ' .. table.concat(cycle, ' -> ', 1, length))
579   end
580 end
581 end
582 realcallbacklist[name] = list
583 t[name] = list
584 return list
585 end
586 }

```

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

```

587 local list, data, exclusive, simple, reverselist = 1, 2, 3, 4, 5
588 local types    = {

```

```

589   list      = list,
590   data      = data,
591   exclusive = exclusive,
592   simple    = simple,
593   reverselist = reverselist,
594 }

```

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.)

```
595 local callbacktypes = callbacktypes or {
```

Section 8.2: file discovery callbacks.

```

596   find_read_file      = exclusive,
597   find_write_file     = exclusive,
598   find_font_file      = data,
599   find_output_file    = data,
600   find_format_file    = data,
601   find_vf_file        = data,
602   find_map_file       = data,
603   find_enc_file       = data,
604   find_pk_file        = data,
605   find_data_file      = data,
606   find_opentype_file  = data,
607   find_truetype_file  = data,
608   find_type1_file     = data,
609   find_image_file     = data,

610   open_read_file      = exclusive,
611   read_font_file      = exclusive,
612   read_vf_file        = exclusive,
613   read_map_file       = exclusive,
614   read_enc_file       = exclusive,
615   read_pk_file        = exclusive,
616   read_data_file      = exclusive,
617   read_truetype_file  = exclusive,
618   read_type1_file     = exclusive,
619   read_opentype_file  = exclusive,

```

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

```

620   find_cidmap_file   = data,
621   read_cidmap_file   = exclusive,

```

Section 8.3: data processing callbacks.

```

622   process_input_buffer = data,
623   process_output_buffer = data,
624   process_jobname      = data,

```

Section 8.4: node list processing callbacks.

```
625 contribute_filter      = simple,
626 buildpage_filter       = simple,
627 build_page_insert      = exclusive,
628 pre_linebreak_filter   = list,
629 linebreak_filter        = exclusive,
630 append_to_vlist_filter = exclusive,
631 post_linebreak_filter  = reverselist,
632 hpack_filter           = list,
633 vpack_filter           = list,
634 hpack_quality          = exclusive,
635 vpack_quality          = exclusive,
636 pre_output_filter      = list,
637 process_rule            = exclusive,
638 hyphenate               = simple,
639 ligaturing              = simple,
640 kerning                 = simple,
641 insert_local_par        = simple,
642 % mlist_to_hlist        = exclusive,
643 new_graf                = exclusive,
```

Section 8.5: information reporting callbacks.

```
644 pre_dump               = simple,
645 start_run               = simple,
646 stop_run                = simple,
647 start_page_number        = simple,
648 stop_page_number         = simple,
649 show_error_hook          = simple,
650 show_warning_message     = simple,
651 show_error_message       = simple,
652 show_lua_error_hook      = simple,
653 start_file               = simple,
654 stop_file                = simple,
655 call_edit                = simple,
656 finish_synctex          = simple,
657 wrapup_run               = simple,
```

Section 8.6: PDF-related callbacks.

```
658 finish_pdffile          = data,
659 finish_pdfpage           = data,
660 page_objnum_provider     = data,
661 page_order_index          = data,
662 process_pdf_image_content = data,
```

Section 8.7: font-related callbacks.

```
663 define_font              = exclusive,
664 glyph_info                = exclusive,
665 glyph_not_found           = exclusive,
666 glyph_stream_provider     = exclusive,
667 make_extensible            = exclusive,
668 font_descriptor_objnum_provider = exclusive,
669 input_level_string         = exclusive,
670 provide_charproc_data     = exclusive,
671 }
672 luatexbase.callbacktypes=callbacktypes
```

Sometimes multiple callbacks correspond to a single underlying engine level callback. Then the engine level callback should be registered as long as at least one of these callbacks is in use. This is implemented through a shared table which counts how many of the involved callbacks are currently in use. The engine level callback is registered iff this count is not 0.

We add `mlist_to_hlist` directly to the list to demonstrate this, but the handler gets added later when it is actually defined.

All callbacks in this list are treated as user defined callbacks.

```

673 local shared_callbacks = {
674   mlist_to_hlist = {
675     callback = "mlist_to_hlist",
676     count = 0,
677     handler = nil,
678   },
679 }
680 shared_callbacks.pre_mlist_to_hlist_filter = shared_callbacks.mlist_to_hlist
681 shared_callbacks.post_mlist_to_hlist_filter = shared_callbacks.mlist_to_hlist

```

`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.

```

682 local callback_register = callback_register or callback.register
683 function callback.register()
684   luatexbase_error("Attempt to use callback.register() directly\n")
685 end

```

(End of 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.

```
686 local function data_handler(name)
687     return function(data, ...)
688         for _,i in ipairs(callbacklist[name]) do
689             data = i.func(data,...)
690         end
691     return data
692 end
693 end
```

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

```
694 local function data_handler_default(value)
695     return value
696 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.

```
697 local function exclusive_handler(name)
698     return function(...)
699         return callbacklist[name][1].func(...)
700     end
701 end
```

Handler for `list` callbacks.

```
702 local function list_handler(name)
703     return function(head, ...)
704         local ret
705         for _,i in ipairs(callbacklist[name]) do
706             ret = i.func(head, ...)
707             if ret == false then
708                 luatexbase_warning(
709                     "Function '" .. i.description .. "' returned false\n"
710                     .. "in callback '" .. name .. "'")
711             )
712             return false
713         end
714         if ret ~= true then
715             head = ret
716         end
717     end
718     return head
719 end
720 end
```

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

```
721 local function list_handler_default(head)
722     return head
723 end
```

Handler for `reverselist` callbacks.

```
724 local function reverselist_handler(name)
725   return function(head, ...)
726     local ret
727     local callbacks = callbacklist[name]
728     for i = #callbacks, 1, -1 do
729       local cb = callbacks[i]
730       ret = cb.func(head, ...)
731       if ret == false then
732         luatexbase_warning(
733           "Function '" .. cb.description .. "' returned false\n"
734           .. "in callback '" .. name .. "'")
735       end
736       return false
737     end
738     if ret ~= true then
739       head = ret
740     end
741   end
742   return head
743 end
744 end
```

Handler for `simple` callbacks.

```
745 local function simple_handler(name)
746   return function(...)
747     for _,i in ipairs(callbacklist[name]) do
748       i.func(...)
749     end
750   end
751 end
```

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

```
752 local function simple_handler_default()
753 end
```

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

```
754 local handlers = {
755   [data]      = data_handler,
756   [exclusive] = exclusive_handler,
757   [list]      = list_handler,
758   [reverselist] = reverselist_handler,
759   [simple]    = simple_handler,
760 }
761 local defaults = {
762   [data]      = data_handler_default,
763   [exclusive] = nil,
764   [list]      = list_handler_default,
765   [reverselist] = list_handler_default,
766   [simple]    = simple_handler_default,
767 }
```

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.

```
768 local user_callbacks_defaults = {}
```

`create_callback` The allocator itself.

```
769 local function create_callback(name, ctype, default)
770   local ctype_id = types[ctype]
771   if not name or name == ""
772   or not ctype_id
773   then
774     luatexbase_error("Unable to create callback:\n" ..
775                      "valid callback name and type required")
776   end
777   if callbacktypes[name] then
778     luatexbase_error("Unable to create callback '" .. name ..
779                      "':\ncallback is already defined")
780   end
781   default = default or defaults[ctype_id]
782   if not default then
783     luatexbase_error("Unable to create callback '" .. name ..
784                      "'\ndefault is required for '" .. ctype ..
785                      "' callbacks")
786   elseif type (default) ~= "function" then
787     luatexbase_error("Unable to create callback '" .. name ..
788                      "':\ndefault is not a function")
789   end
790   user_callbacks_defaults[name] = default
791   callbacktypes[name] = ctype_id
792 end
793 luatexbase.create_callback = create_callback
```

(End of definition for `create_callback`.)

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

```
794 local function call_callback(name,...)
795   if not name or name == "" then
796     luatexbase_error("Unable to create callback:\n" ..
797                      "valid callback name required")
798   end
799   if user_callbacks_defaults[name] == nil then
800     luatexbase_error("Unable to call callback '" .. name ..
801                      "':\nunknown or empty")
802   end
803   local l = callbacklist[name]
804   local f
805   if not l then
806     f = user_callbacks_defaults[name]
807   else
808     f = handlers[callbacktypes[name]](name)
809   end
810   return f(...)
```

```

811 end
812 luatexbase.call_callback=call_callback

```

(End of definition for `call_callback`.)

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

```

813 local function add_to_callback(name, func, description)
814     if not name or name == "" then
815         luatexbase_error("Unable to register callback:\n" ..
816                         "valid callback name required")
817     end
818     if not callbacktypes[name] or
819         type(func) ~= "function" or
820         not description or
821         description == "" then
822         luatexbase_error(
823             "Unable to register callback.\n\n"
824             .. "Correct usage:\n"
825             .. "add_to_callback(<callback>, <function>, <description>)"
826         )
827     end

```

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

```

828 local l = realcallbacklist[name]
829 if l == nil then
830     l = { }
831     realcallbacklist[name] = l

```

Handle count for shared engine callbacks.

```

832 local shared = shared_callbacks[name]
833 if shared then
834     shared.count = shared.count + 1
835     if shared.count == 1 then
836         callback_register(shared.callback, shared.handler)
837     end

```

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

```

838 elseif user_callbacks_defaults[name] == nil then
839     callback_register(name, handlers[callbacktypes[name]](name))
840 end
841 end

```

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

```

842 local f = {
843     func      = func,
844     description = description,
845 }
846 if callbacktypes[name] == exclusive then
847     if #l == 1 then
848         luatexbase_error(
849             "Cannot add second callback to exclusive function\n" ..
850             name .. "'")
851     end
852 end

```

```

853     table.insert(l, f)
854     callbacklist[name] = nil
855     luatexbase_log(
856         "Inserting '" .. description .. "' in '" .. name .. "'."
857     )
858 end
859 luatexbase.add_to_callback = add_to_callback

(End of definition for add_to_callback.)

declare_callback_rule Add an ordering constraint between two callback implementations
860 local function declare_callback_rule(name, desc1, relation, desc2)
861     if not callbacktypes[name] or
862         not desc1 or not desc2 or
863         desc1 == "" or desc2 == "" then
864         luatexbase_error(
865             "Unable to create ordering constraint. "
866             .. "Correct usage:\n"
867             .. "declare_callback_rule(<callback>, <description_a>, <description_b>)"
868         )
869     end
870     if relation == 'before' then
871         relation = nil
872     elseif relation == 'after' then
873         desc2, desc1 = desc1, desc2
874         relation = nil
875     elseif relation == 'incompatible-warning' or relation == 'incompatible-error' then
876     elseif relation == 'unrelated' then
877     else
878         luatexbase_error(
879             "Unknown relation type in declare_callback_rule"
880         )
881     end
882     callbacklist[name] = nil
883     local rules = callbackrules[name]
884     if rules then
885         for i, rule in ipairs(rules) do
886             if rule[1] == desc1 and rule[2] == desc2 or rule[1] == desc2 and rule[2] == desc1 then
887                 if relation == 'unrelated' then
888                     table.remove(rules, i)
889                 else
890                     rule[1], rule[2], rule.type = desc1, desc2, relation
891                 end
892                 return
893             end
894         end
895         if relation ~= 'unrelated' then
896             rules[#rules + 1] = {desc1, desc2, type = relation}
897         end
898     elseif relation ~= 'unrelated' then
899         callbackrules[name] = {{desc1, desc2, type = relation}}
900     end
901 end

```

```

902 luatexbase.declare_callback_rule = declare_callback_rule
(End of definition for declare_callback_rule.)

remove_from_callback Remove a function from a callback. First check arguments.
903 local function remove_from_callback(name, description)
904   if not name or name == "" then
905     luatexbase_error("Unable to remove function from callback:\n" ..
906                       "valid callback name required")
907   end
908   if not callbacktypes[name] or
909       not description or
910       description == "" then
911     luatexbase_error(
912       "Unable to remove function from callback.\n\n"
913       .. "Correct usage:\n"
914       .. "remove_from_callback(<callback>, <description>)"
915     )
916   end
917   local l = realcallbacklist[name]
918   if not l then
919     luatexbase_error(
920       "No callback list for " .. name .. "'\n")
921   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.
922   local index = false
923   for i,j in ipairs(l) do
924     if j.description == description then
925       index = i
926       break
927     end
928   end
929   if not index then
930     luatexbase_error(
931       "No callback " .. description .. " registered for " ..
932       name .. "'\n")
933   end
934   local cb = l[index]
935   table.remove(l, index)
936   luatexbase_log(
937     "Removing " .. description .. " from " .. name .. "."
938   )
939   if #l == 0 then
940     realcallbacklist[name] = nil
941     callbacklist[name] = nil
942     local shared = shared_callbacks[name]
943     if shared then
944       shared.count = shared.count - 1
945       if shared.count == 0 then
946         callback_register(shared.callback, nil)
947       end
948     elseif user_callbacks_defaults[name] == nil then
949       callback_register(name, nil)

```

```

950     end
951   end
952   return cb.func,cb.description
953 end
954 luatexbase.remove_from_callback = remove_from_callback

(End of definition for remove_from_callback.)

```

in_callback Look for a function description in a callback.

```

955 local function in_callback(name, description)
956   if not name
957     or name == ""
958     or not realcallbacklist[name]
959     or not callbacktypes[name]
960     or not description then
961       return false
962     end
963   for _, i in pairs(realcallbacklist[name]) do
964     if i.description == description then
965       return true
966     end
967   end
968   return false
969 end
970 luatexbase.in_callback = in_callback

```

(*End of definition for in_callback.*)

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

```

971 local function disable_callback(name)
972   if(realcallbacklist[name] == nil) then
973     callback_register(name, false)
974   else
975     luatexbase_error("Callback list for " .. name .. " not empty")
976   end
977 end
978 luatexbase.disable_callback = disable_callback

```

(*End of definition for disable_callback.*)

callback_descriptions List the descriptions of functions registered for the given callback. This will sort the list if necessary.

```

979 local function callback_descriptions (name)
980   local d = {}
981   if not name
982     or name == ""
983     or not realcallbacklist[name]
984     or not callbacktypes[name]
985     then
986       return d
987   else
988     for k, i in pairs(callbacklist[name]) do
989       d[k]= i.description
990     end
991   end

```

```

992     return d
993 end
994 luatexbase.callback_descriptions =callback_descriptions

(End of 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!

```

995 local function uninstall()
996   module_info(
997     "luatexbase",
998     "Uninstalling kernel luatexbase code"
999   )
1000   callback.register = callback_register
1001   luatexbase = nil
1002 end
1003 luatexbase.uninstall = uninstall
```

(End of 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.

```

1004 create_callback('pre_mlist_to_hlist_filter', 'list')
1005 create_callback('mlist_to_hlist', 'exclusive', node.mlist_to_hlist)
1006 create_callback('post_mlist_to_hlist_filter', 'list')
1007 function shared_callbacks.mlist_to_hlist.handler(head, display_type, need_penalties)
1008   local current = call_callback("pre_mlist_to_hlist_filter", head, display_type, need_penalties)
1009   if current == false then
1010     flush_list(head)
1011     return nil
1012   end
1013   current = call_callback("mlist_to_hlist", current, display_type, need_penalties)
1014   local post = call_callback("post_mlist_to_hlist_filter", current, display_type, need_penalties)
1015   if post == false then
1016     flush_list(current)
1017     return nil
1018   end
1019   return post
1020 end
```

(End of definition for `mlist_to_hlist`.)

1021 `/lua`

Reset the catcode of `Ø`.

1022 `\tex\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 of definition for \@kernel@after@enddocument and others.)

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

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

Create a hook for last-minute expl3 material.

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

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

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

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

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

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

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

```

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

```

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

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

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

```

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

```

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

```

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

```

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

```

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

```

```

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

```

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

```
132 〈latexrelease〉\@ifundefined{ExplSyntaxOff}{}{\textrm{latexrelease}\@postltxexpl}
```

1.2 Using `expl3` code

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

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

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

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

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

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

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

```

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

```

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

```

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

```

```

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

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

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

```

2 Document-level command names for `expl3` functions

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

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

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

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

```

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

```

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

```

160 \cs_new_eq:NN \inteval \int_eval:n

```

```

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

(End of definition for \fpeval and others.)

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

```

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

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

```

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

```

(End of definition for \UserName and \ExpandArgs.)

```

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

```

File f

ltdfns.dtx

1 Definitions

This section contains commands used in defining other macros.

1 `(*2ekernel)`

1.1 Initex initializations

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

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

(*End of 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 of definition for \typeout.*)

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

20 `\newlinechar`\^J`

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

\@@hyph Save original primitive definition.
²⁴ \let\@@hyph=\-

(End of definition for \@@hyph.)

\@@italiccorr Save the original italic correction.
²⁵ \let\@@italiccorr=\/

(End of definition for \@@italiccorr.)

\@height The following definitions save token space. E.g., using \@height instead of height saves
\@depth 5 tokens at the cost in time of one macro expansion.

\@width ²⁶ \def\@height{height} \def\@depth{depth} \def\@width{width}

\@minus ²⁷ \def\@minus{minus}

\@plus ²⁸ \def\@plus{plus}

The next one is another 100 tokens worth.
²⁹ \def\hb@xt@{\hbox to}

(End of definition for \@height and others.)

³⁰ \message{hacks,}

\hb@xt@

1.3 Command definitions

This section defines the following commands:

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

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

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

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

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

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

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

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

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

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

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

`\renewenvironment` Obvious companion to `\newenvironment`.
`\@cons` : See description of `\output` routine.
`\@car` $\@car T_1 T_2 \dots T_n \@nil == T_1$ (unexpanded)
`\@cdr` $\@cdr T_1 T_2 \dots T_n \@nil == T_2 \dots T_n$ (unexpanded)
`\typeout` $\{\langle message \rangle\}$
 Produces a warning message on the terminal.
`\typein` $\{\langle message \rangle\}$
 Types message, asks the user to type in a command, then executes it
`\typein` $[(\text{\texttt{CS}})]\{\langle MSG \rangle\}$
 Same as above, except defines `\text{\texttt{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 of definition for `\typein`.)

`\@namedef`

```

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

```

(End of definition for `\@namedef`.)

`\@nameuse`

```

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

```

(End of definition for `\@nameuse`.)

```

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

```

\@car
\@cdr 53 \def\@car#1#2\@nil{#1}
54 \def\@cdr#1#2\@nil{#2}
(End of 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 of definition for \@carcube.)
```

\@onlypreamble 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 of 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 of 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 of 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 of definition for \newcommand and \new@command.)

```

\@newcommand Handling arguments for \newcommand.

```

\@argdef 80 \def\@newcommand#1[#2]{%
\@xargdef 81 \kernel@ifnextchar [{\@xargdef#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 of 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 of 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 of 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 of definition for \@yargdef and \@yargd@f.)

\@reargdef

```

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

```

(End of definition for \@reargdef.)

- \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 of 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 of 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]}}
152 \def\renewenvironment{\@star@or@long\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}}
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}}
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@}

```

(End of definition for `\newenvironment` and `\renewenvironment`.)

`\@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 by temporarily letting the undefined `\...` (begin code) to the definition of `\end...` and as a result we get an error if that has a definition.

(End of 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 of 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 of 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 of 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 of 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 of definition for \@check@eq.)
```

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

```

208 \long\def \@gobble #1{}
\gobbletwo \long\def \@gobbletwo #1#2{}
\gobblethree \long\def \@gobblethree #1#2#3{}
\gobblefour \long\def \@gobblefour #1#2#3#4{}

(End of definition for \@gobble and others.)
```

\@firstofone Some argument-grabbers.

```

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

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

(End of definition for \@firstofone and others.)
```

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

```

216 \long\def \@thirddofthree#1#2#3{#3}

(End of definition for \@thirddofthree.)
```

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

```

217 </2ekernel>
218 <latexrelease>\IncludeInRelease{2022/11/01}%
219 <latexrelease>      {\@expandtwoargs}{protected edef}%
220 <*2ekernel | latexrelease>
221 \def\@expandtwoargs#1#2#3{%
222   \protected@edef\reserved@a{\noexpand#1{#2}{#3}}\reserved@a}
223 </2ekernel | latexrelease>
224 <latexrelease>\EndIncludeInRelease
225 <latexrelease>\IncludeInRelease{00/00/00}%
226 <latexrelease>      {\@expandtwoargs}{protected edef}%
227 <latexrelease>\def\@expandtwoargs#1#2#3{%
228   \def\@expandtwoargs#1#2#3{\noexpand#1{#2}{#3}}\reserved@a}
229 <latexrelease>\EndIncludeInRelease
230 <*2ekernel>
```

(End of definition for \@expandtwoargs.)

\@backslashchar A category code 12 backslash.

²³¹ \edef\@backslashchar{\expandafter\gobble\string\\}

(End of 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.
- \cunexpandable@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.

\cunexpandable@protect

²³² \def\cunexpandable@protect{\noexpand\protect\noexpand}

(End of definition for \cunexpandable@protect.)

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

\DeclareRobustCommand\foo

define \foo to be \protect\foo<space>,

and then use \newcommand\foo<space>.

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

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

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

which expands to:

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

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

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

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

```
233 \def\DeclareRobustCommand{\@star@or@long\declare@robustcommand}  
234 \def\declare@robustcommand#1{  
235   \ifx#1\undefined\else\ifx#1\relax\else  
236     \@latex@info{Redefining \string#1}  
237   \fi\fi  
238   \edef\reserved@a{\string#1}  
239   \def\reserved@b{\string#1}  
240   \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}  
241   \edef#1{  
242     \ifx\reserved@a\reserved@b  
243       \noexpand\x@protect  
244       \noexpand#1  
245     \fi  
246     \noexpand\protect  
247     \expandafter\noexpand\csname  
248       \expandafter\gobble\string#1 \endcsname  
249   }%  
250   \let\@ifdefinable\@rc@ifdefinable  
251   \expandafter\new@command\csname  
252     \expandafter\gobble\string#1 \endcsname  
253 }
```

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

```
\x@protect  
254 \def\x@protect#1{  
255   \ifx\protect\@typeset@protect\else  
256     \x@protect#1  
257   \fi  
258 }  
259 \def\x@protect#1\fi#2#3{  
260   \fi\protect#1  
261 }
```

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

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

```
262 \let\@typeset@protect\relax
```

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

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

```
263 \def\set@display@protect{\let\protect\string}
264 \def\set@typeset@protect{\let\protect\@typeset@protect}
```

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

\protected@edef \protected@xdef \unrestored@protected@xdef \restore@protect 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!

```
265 \def\protected@edef{%
266   \let\@@protect\protect
267   \let\protect\@unexpandable@protect
268   \afterassignment\restore@protect
269   \edef
270 }
271 \def\protected@xdef{%
272   \let\@@protect\protect
273   \let\protect\@unexpandable@protect
274   \afterassignment\restore@protect
275   \xdef
276 }
277 \def\unrestored@protected@xdef{%
278   \let\protect\@unexpandable@protect
279   \xdef
280 }
281 \def\restore@protect{\let\protect\@@protect}
```

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

\protect The normal meaning of \protect

```
282 \set@typeset@protect
```

(End of 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.

```
283 </2ekernel>
```

```
284 <latexrelease>\IncludeInRelease{2020/10/01}{\MakeRobust}{\MakeRobust}%
```

```

285  {*2ekernel | latexrelease}
286  \def\MakeRobust#1{%
287    \count@=\escapechar
288    \escapechar='\\
289    \@ifundefined{\expandafter\gobble\string#1}{%
290      \@latex@error{Command '\string#1' undefined.%}
291      \MessageBreak There is nothing here to make robust}%
292    \e@ha
293  }%

```

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.

```

294  {%
295  \@ifundefined{\expandafter\gobble\string#1\space}{%
296    {%
297      \expandafter\@kernel@rename@newcommand
298      \csname\expandafter\gobble\string#1\space\endcsname
299      #1%
300      \edef\reserved@a{\string#1}%
301      \def\reserved@b{#1}%
302      \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
303      \xdef#1{%
304        \ifx\reserved@a\reserved@b
305          \noexpand\x@protect\noexpand#1%
306        \fi
307        \noexpand\protect\expandafter\noexpand
308        \csname\expandafter\gobble\string#1\space\endcsname}%
309    }%
310    {\@latex@info{Command '\string#1' is already robust}}%
311  }%
312  \escapechar=\count@
313 }%

```

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.

```

314  \def\@kernel@rename@newcommand#1#2{%
315    \robust@command@chk@safe#2%
316    {\@if@newcommand#2%
317      {\afterassignment\global
318        \global\copy@newcommand#1#2%
319        \global\let#2@\undefined

```

```

320      \global\expandafter\let\csname string#2\endcsname\@undefined}%
321      {\global\let#1=#2}%%
322      {\global\let#1=#2}

323  (/2ekernel | latexrelease)
324  <latexrelease>\EndIncludeInRelease
325  %
326  <latexrelease>\IncludeInRelease{2019/10/01}{\MakeRobust}{\MakeRobust}%
327  <latexrelease>\def\MakeRobust#1{%
328  <latexrelease>  \@ifundefined{\expandafter\gobble\string#1}{%
329  <latexrelease>    \@latex@error{The control sequence ‘\string#1’ is undefined!}%
330  <latexrelease>      \MessageBreak There is nothing here to make robust}%
331  <latexrelease>    \c@eha
332  <latexrelease> }%
333  <latexrelease> }%
334  <latexrelease>  \@ifundefined{\expandafter\gobble\string#1\space}{%
335  <latexrelease>    }%
336  <latexrelease>    \global\expandafter\let\csname
337  <latexrelease>      \expandafter\gobble\string#1\space\endcsname=#1%
338  <latexrelease>      \edef\reserved@a{\string#1}%
339  <latexrelease>      \def\reserved@b{#1}%
340  <latexrelease>      \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
341  <latexrelease>      \xdef#1{%
342  <latexrelease>        \ifx\reserved@a\reserved@b
343  <latexrelease>          \noexpand\x@protect\noexpand#1%
344  <latexrelease>        \fi
345  <latexrelease>        \noexpand\protect\expandafter\noexpand
346  <latexrelease>        \csname\expandafter\gobble\string#1\space\endcsname}%
347  <latexrelease> }%
348  <latexrelease>  {\@latex@info{The control sequence ‘\string#1’ is already robust}}%
349  <latexrelease> }%
350  <latexrelease>}%
351  <latexrelease>\let\@kernel@rename@newcommand\@undefined
352  <latexrelease>\EndIncludeInRelease
353  %
354  <latexrelease>\IncludeInRelease{2015/01/01}{\MakeRobust}{\MakeRobust}%
355  <latexrelease>\def\MakeRobust#1{%
356  <latexrelease>  \@ifundefined{\expandafter\gobble\string#1}{%
357  <latexrelease>    \@latex@error{The control sequence ‘\string#1’ is undefined!}%
358  <latexrelease>      \MessageBreak There is nothing here to make robust}%
359  <latexrelease>    \c@eha
360  <latexrelease> }%
361  <latexrelease> }%
362  <latexrelease>  \@ifundefined{\expandafter\gobble\string#1\space}{%
363  <latexrelease>    }%
364  <latexrelease>    \expandafter\let\csname
365  <latexrelease>      \expandafter\gobble\string#1\space\endcsname=#1%
366  <latexrelease>      \edef\reserved@a{\string#1}%
367  <latexrelease>      \def\reserved@b{#1}%
368  <latexrelease>      \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
369  <latexrelease>      \xdef#1{%
370  <latexrelease>        \ifx\reserved@a\reserved@b
371  <latexrelease>          \noexpand\x@protect\noexpand#1%
372  <latexrelease>        \fi

```

```

373 〈\latexrelease〉      \noexpand\protect\expandafter\noexpand
374 〈\latexrelease〉      \csname\expandafter@gobble\string#1\space\endcsname}%
375 〈\latexrelease〉      }%
376 〈\latexrelease〉      {\@latex@info{The control sequence ‘\string#1’ is already robust}}%
377 〈\latexrelease〉      }%
378 〈\latexrelease〉}%
379 〈\latexrelease〉\let\@kernel@rename@newcommand\@undefined
380 〈\latexrelease〉\EndIncludeInRelease
381 %
382 〈\latexrelease〉\IncludeInRelease{0000/00/00}{\MakeRobust}{\MakeRobust}%
383 〈\latexrelease〉\let\MakeRobust\@undefined
384 〈\latexrelease〉\let\@kernel@rename@newcommand\@undefined
385 〈\latexrelease〉\EndIncludeInRelease
386 {*ekernel}

```

(End of 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.

```

387 〈/2ekernel〉
388 {*2ekernel | \latexrelease〉
389 〈\latexrelease〉\IncludeInRelease{2020/10/01}%
390 〈\latexrelease〉          {\@kernel@make@fragile}{Undo robustness}%
391 \def\kernel@make@fragile#1{%
392   \@ifundefined{\expandafter\@gobble\string#1\space}%

```

If not robust do nothing.

```
393   {}%

```

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_`.

```

394   {}%
395   \global\expandafter\let\expandafter #1\csname
396     \expandafter\@gobble\string#1\space\endcsname
397   \expandafter\@kernel@rename@newcommand
398     \csname\expandafter\@gobble\string#1\expandafter\endcsname
399     \csname\expandafter\@gobble\string#1\space\endcsname
400   \global\expandafter\let\csname
401     \expandafter\@gobble\string#1\space\endcsname\@undefined
402   }%
403 }
404 〈\latexrelease〉\EndIncludeInRelease
405 %
406 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
407 〈\latexrelease〉          {\@kernel@make@fragile}{Undo robustness}%
408 〈\latexrelease〉\def\kernel@make@fragile#1{%
409 〈\latexrelease〉  \@ifundefined{\expandafter\@gobble\string#1\space}%
410 〈\latexrelease〉    {}%
411 〈\latexrelease〉    {}%

```

```
412 \langle\latextrelease\rangle      \global\expandafter\let\expandafter #1\csname
413 \langle\latextrelease\rangle      \expandafter\@gobble\string#1\space\endcsname
414 \langle\latextrelease\rangle      \global\expandafter\let\csname
415 \langle\latextrelease\rangle      \expandafter\@gobble\string#1\space\endcsname\@undefined
416 \langle\latextrelease\rangle    }%
417 \langle\latextrelease\rangle}
418 \langle\latextrelease\rangle\EndIncludeInRelease
419 \langle/2ekernel | \latextrelease\rangle
420 \langle*2ekernel\rangle
```

(End of definition for \kernel@make@fragile.)

1.5 Acting on robust commands

```
421 </2ekernel>
422 <latexrelease>\IncludeInRelease{2020-10-01}{\robust@command@act}
423 <latexrelease> {Add \robust@command@act}%
424 <*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 `\LaTeX`'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

\robust@command@act{action-list}{robust-cmd}

Wattbach weiter / weiter

{ $\langle if-type-1 \rangle$ $\langle act-type-1 \rangle$ }
{ $\langle if-type-2 \rangle$ $\langle act-type-2 \rangle$ }

`\robust@command@act` will iterate over the $\langle action-list \rangle$, evaluating each $\langle if-type-n \rangle \langle robust-cmd \rangle \{ \langle true \rangle \} \{ \langle false \rangle \}$. If the $\langle if-type-n \rangle$ conditional returns $\langle true \rangle$, then $\langle act-type-n \rangle \langle act-arg \rangle$ is executed, and the loop ends. If the conditional returns $\langle false \rangle$, then $\langle if-type-n + 1 \rangle$ is executed in the same way, until either one of the conditionals return $\langle true \rangle$, or the end of the $\langle action-list \rangle$ is reached. If the end is reached, then $\langle fallback-action \rangle \langle act-arg \rangle$ 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.

```

425 \long\def\robust@command@act#1#2#3#4{%
426   \robust@command@chk@safe#2%
427   {\expandafter\robust@command@act@loop
428     \expandafter#2%
429     #1{\@nnnil\@nnnil}%
430   \robust@command@act@end}%
431   {\robust@command@act@end}%
432   {#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.

```

433 \long\def\robust@command@act@loop#1#2{\robust@command@act@loop@aux#1#2}
434 \long\def\robust@command@act@loop@aux#1#2#3{%
435   \ifx\@nnil#2%
436   \else
437     #2{#1}%
438     {\robust@command@act@do{#3}}%
439     {\expandafter\robust@command@act@loop\expandafter#1}%
440   \fi}
441 \long\def\robust@command@act@do#1%
442 \fi#2%
443 \robust@command@act@end#3#4{%
444 \fi
445 #1#4}

```

If the end is reached and no action was taken, then do *<fallback-action><act-arg>*.

```
\robust@command@act@end 446 \long\def\robust@command@act@end#1#2{#1#2}
```

```

447 \long\def\robust@command@chk@safe#1{%
448   \begingroup
449   \escapechar='\\
450   \expandafter\endgroup\expandafter
451   \robust@command@act@chk@args\meaning#1:->\@nil}%
452 \def\robust@command@act@chk@args#1:->#2\@nil{%
453   \expl@str@if@eq@@nnTF{#1}{macro}%
454   {\@firstoftwo}%
455   {\expl@str@if@eq@@nnTF{#1}{\protected macro}%
456   {\@firstoftwo}%
457   {\@secondoftwo}}}%
458 (/2ekernel | latexrelease)
459 (latexrelease)\EndIncludeInRelease
460 (latexrelease)\IncludeInRelease{0000-00-00}\robust@command@act}
461 (latexrelease) {Add \robust@command@act}%

```

```

462 〈\latexrelease〉\let\robust@command@act\@undefined
463 〈\latexrelease〉\let\robust@command@act@loop\@undefined
464 〈\latexrelease〉\let\robust@command@act@loop@aux\@undefined
465 〈\latexrelease〉\let\robust@command@act@do\@undefined
466 〈\latexrelease〉\let\robust@command@act@end\@undefined
467 〈\latexrelease〉\let\robust@command@chk@safe\@undefined
468 〈\latexrelease〉\let\robust@command@act@chk@args\@undefined
469 〈\latexrelease〉\EndIncludeInRelease
470 {*2ekernel}

```

(End of definition for \robust@command@act and others.)

1.5.1 Copying robust commands

```

471 〈/2ekernel〉
472 〈\latexrelease〉\IncludeInRelease{2020-10-01}{\DeclareCommandCopy}
473 〈\latexrelease〉 {Add \NewCommandCopy, \RenewCommandCopy, and \DeclareCommandCopy}%
474 {*2ekernel | latexrelease}

```

\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.

```

475 \def\NewCommandCopy{%
476   \declare@commandcopy
477   {\@firstofone}%
478   {\@firstoftwo\@notdefinable}}%
479 \def\RenewCommandCopy{%
480   \declare@commandcopy
481   {\@latex@error{Command \@backslashchar\reserved@a\space undefined}\@ehc
482   \@firstofone}%
483   {\@firstofone}}%
484 \def\DeclareCommandCopy{%
485   \declare@commandcopy
486   {\@firstofone}%
487   {\@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>*.

```

488 \long\def\declare@commandcopy#1#2#3#4{%
489   \edef\reserved@a{\@expl@cs@to@str@@N#3}%
490   \@ifundefined\reserved@a{#1}{#2}%
491     {\declare@commandcopy@do{#3}{#4}}}
492 \long\def\declare@commandcopy@do#1#2{%
493   \robust@command@act
494   \@declarecommandcopylisthook#2%
495   \declare@commandcopy@let{#1#2}}

```

The initial definition of `\@declarecommandcopylisthook` contains the tests for the two types of robust command in the kernel.

```

496 \def\@declarecommandcopylisthook{%
497   {\@if@DeclareRobustCommand \@copy@DeclareRobustCommand}%
498   {\@if@newcommand \@copy@newcommand}}

```

The initial definition of `\@declarecommandcopylisthook` contains the tests for the two types of robust command in the kernel.

```
499 \long\def\declare@commandcopy@let#1#2{\let#1=#2\relax}
```

`\declare@commandcopy@let`

Now the rollback code.

```

500 {/2ekernel | latexrelease}
501 <latexrelease>\EndIncludeInRelease
502 <latexrelease>\IncludeInRelease{0000-00-00}{\DeclareCommandCopy}
503 <latexrelease> {Undefine \NewCommandCopy, \RenewCommandCopy, and \DeclareCommandCopy}%
504 <latexrelease>\let\NewCommandCopy\@undefined
505 <latexrelease>\let\RenewCommandCopy\@undefined
506 <latexrelease>\let\DeclareCommandCopy\@undefined
507 <latexrelease>\let\declare@commandcopy\@undefined
508 <latexrelease>\let\@declarecommandcopylisthook\@undefined
509 <latexrelease>\let\declare@commandcopy@let\@undefined
510 <latexrelease>\EndIncludeInRelease
511 {*2ekernel}

```

(End of definition for `\NewCommandCopy` and others.)

```

512 {/2ekernel}
513 <latexrelease>\IncludeInRelease{2023-06-01}{\DeclareEnvironmentCopy}
514 <latexrelease> {Add \NewEnvironmentCopy, \RenewEnvironmentCopy, and \DeclareEnvironmentCopy}%
515 {*2ekernel | latexrelease}

```

`\NewEnvironmentCopy` If `\#1` or `\end#1` already exist one gets an error message talking about the problematical command (not the environment). The remainder of the L^AT_EX run is probably badly broken and it is unlikely that continuing it gives reasonable results.

```

516 \def\NewEnvironmentCopy{%
517   \declare@environmentcopy
518   {\@firstofone}%
519   {\@firstoftwo\@notdefinable}%
520 \def\RenewEnvironmentCopy{%
521   \declare@environmentcopy
522   {\@latex@error{Environment \reserved@a\space undefined}\@ehc
523   {\@firstofone}%
524   {\@firstofone}}}

```

```

525 \def\DeclareEnvironmentCopy{%
526   \declare@environmentcopy
527   {\@firstofone}%
528   {\@firstofone}%
529 \long\def\declare@environmentcopy#1#2#3#4{%
530   \edef\reserved@a{\@ifundefined{#3}{\end#3}{#3}}%
531   \@ifundefined\reserved@a
532     {\def\reserved@a{#3}#1}%
533     {\def\reserved@a{#3}#2}%
534   {\ExpandArgs{cc}\declare@commandcopy@do{#3}{#4}}%
535   {\ExpandArgs{cc}\declare@commandcopy@do{\end#3}{\end#4}}}%

```

Now the rollback code.

```

536 </2ekernel | latexrelease>
537 <latexrelease>\EndIncludeInRelease
538 <latexrelease>\IncludeInRelease{0000-00-00}{\DeclareEnvironmentCopy}
539 <latexrelease> {\Undefine \NewEnvironmentCopy, \RenewEnvironmentCopy, and \DeclareEnvironmentCopy}
540 <latexrelease>\let\NewEnvironmentCopy\@undefined
541 <latexrelease>\let\RenewEnvironmentCopy\@undefined
542 <latexrelease>\let\DeclareEnvironmentCopy\@undefined
543 <latexrelease>\EndIncludeInRelease
544 <*2ekernel>

```

(*End of definition for \NewEnvironmentCopy , \RenewEnvironmentCopy , and \DeclareEnvironmentCopy.*)

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.

```

545 </2ekernel>
546 <latexrelease>\IncludeInRelease{2020-10-01}{\ShowCommand}%
547 <latexrelease> {\Add \ShowCommand}%
548 <*2ekernel | latexrelease>

549 \long\def\ShowCommand#1{%
550   \robust@command@act
551   \showcommandlisthook#1%
552   \show#1}

```

- \showcommandlisthook 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.

```

553 \def\showcommandlisthook{%
554   {\@if@DeclareRobustCommand \show@DeclareRobustCommand}%
555   {\@if@newcommand \show@newcommand}}

```

Now the rollback code.

```

556 </2ekernel | latexrelease>
557 <latexrelease>\EndIncludeInRelease

```

```

558 〈\latexrelease〉\IncludeInRelease{0000-00-00}{\ShowCommand}
559 〈\latexrelease〉 {Undefined \ShowCommand}%
560 〈\latexrelease〉\let\ShowCommand@undefined
561 〈\latexrelease〉\let\@showcommandlisthook@undefined
562 〈\latexrelease〉\EndIncludeInRelease
563 {*2ekernel}

(End of definition for \ShowCommand and \@showcommandlisthook.)
```

564 〈/2ekernel〉

565 〈\latexrelease〉\IncludeInRelease{2020-10-01}{\@if@DeclareRobustCommand}

566 〈\latexrelease〉 {Add \@if@DeclareRobustCommand, \@if@newcommand,

567 〈\latexrelease〉 \copy@DeclareRobustCommand, \copy@newcommand,

568 〈\latexrelease〉 \show@DeclareRobustCommand, \show@newcommand}%

569 {*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.

```

570 \long\def\@if@DeclareRobustCommand#1{%
571   \begingroup
572   \escapechar='\\
573   \edef\reserved@a{\string#1}%
574   \edef\reserved@b{\detokenize{#1}}%
575   \xdef\@gtempa{%
576     \ifx\reserved@a\reserved@b
577       \noexpand\x@protect
578       \noexpand#1%
579     \fi
580     \noexpand\protect
581     \expandafter\noexpand\csname\expl@cs@to@str@N#1 \endcsname}%
582   \endgroup
583   \ifx\@gtempa#1\relax
584     \expandafter\@firstoftwo
585   \else
586     \expandafter\@secondoftwo
587   \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.

```

588 \long\def\copy@DeclareRobustCommand#1#2{%
589   \begingroup
590   \escapechar='\\
591   \edef\reserved@a{\string#1}%
592   \edef\reserved@b{\detokenize{#1}}%
```

```

593      \edef\reserved@a{%
594    \endgroup
595    \def\noexpand#1{%
596      \ifx\reserved@a\reserved@b
597        \noexpand\x@protect
598        \noexpand#1%
599      \fi
600      \noexpand\protect
601      \expandafter\noexpand\csname\@expl@cs@to@str@0N#1 \endcsname}%
602    \noexpand\copy@kernel@robust@command
603    \expandafter\noexpand\csname\@expl@cs@to@str@0N#1 \endcsname
604    \expandafter\noexpand\csname\@expl@cs@to@str@0N#2 \endcsname}%
605  \reserved@a}
606 \long\def\copy@kernel@robust@command#1#2{%
607   \robust@command@chk@safemode#2%
608   {\@if@newcommand#2%
609     {\@copy@newcommand}%
610     {\declare@commandcopy@let}}%
611   {\declare@commandcopy@let}%
612   #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`.

```

613 \long\def\@show@DeclareRobustCommand#1{%
614   \typeout{> \string#1=robust macro:}%
615   \typeout{->@\expl@cs@replacement@spec@0N#1.^~J}%
616   \expandafter\show@kernel@robust@command
617   \csname\@expl@cs@to@str@0N#1 \endcsname}%
618 \long\def\show@kernel@robust@command#1{%
619   \robust@command@chk@safemode#1%
620   {\@if@newcommand#1%
621     {\@show@newcommand}%
622     {\@show@macro}}%
623   {\@show@macro}%
624   #1}
625 \let\@show@macro\show

```

(End of 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.

```

626 \long\def\@if@newcommand#1{%
627   \edef\reserved@a{%
628     \noexpand\@protected@testopt
629     \noexpand#1%
630     \expandafter\noexpand\csname\@backslashchar\expl@cs@to@str@@N#1\endcsname}%
631   \edef\reserved@b{%
632     \unexpanded\expandafter\expandafter\expandafter
633     {\expandafter\@car\@cube#1{}{}{\@nil}}}%
634   \ifx\reserved@a\reserved@b
635     \expandafter\@firstoftwo
636   \else
637     \expandafter\@secondoftwo
638   \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>}.

```

639 \long\def\@copy@newcommand#1#2{%
640   \edef#1{\noexpand\@protected@testopt
641   \noexpand#1%
642   \expandafter\noexpand\csname\@backslashchar\expl@cs@to@str@@N#1\endcsname
643   \unexpanded\expandafter\expandafter\expandafter
644   {\expandafter\@gobblethree#2}}%
645   \expandafter
646   \let\csname\@backslashchar\expl@cs@to@str@@N#1\expandafter\endcsname
647   \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.

```

648 \long\def\@show@newcommand#1{%
649   \typeout{> \string#1=robust macro:}%
650   \typeout{->\@expl@cs@replacement@spec@@N#1.^~J}%
651   \expandafter\@show@newcommand@aux
652   \csname\@backslashchar\expl@cs@to@str@@N#1\expandafter\endcsname
653   \expandafter{\#1}\@show@tokens}

```

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`.

```
654 \long\def\@show@newcommand@aux#1#2#3{%
655   \typeout{> \string#1=\@expl@cs@prefix@spec@@N#1macro:}%
656   #3{default \string##1=\expandafter\detokenize@gobblethree#2.^~J%
657     \@expl@cs@argument@spec@@N#1->\@expl@cs@replacement@spec@@N#1}}
```

This macro prints the contents of the token list (macro) #1 using `\showtokens`. The `\expandafter` gymnastics ensures that `\showtokens` itself, and the internals of this macro aren't showed in the context lines.

```
658 \long\def\@show@tokens#1{%
659   \edef\reserved@a{\#1}%
660   \showtokens\expandafter
661     \expandafter\expandafter{\expandafter\reserved@a}}
```

Now the rollback code.

```
662 </2ekernel | latexrelease>
663 <latexrelease>\EndIncludeInRelease
664 <latexrelease>\IncludeInRelease{0000-00-00}{\@if@DeclareRobustCommand}
665 <latexrelease> {Undefine \@if@DeclareRobustCommand, \@if@newcommand,
666 <latexrelease>           \copy@DeclareRobustCommand, \copy@newcommand,
667 <latexrelease>           \show@DeclareRobustCommand, \show@newcommand}%
668 <latexrelease>\let\@if@DeclareRobustCommand\@undefined
669 <latexrelease>\let\copy@DeclareRobustCommand\@undefined
670 <latexrelease>\let\show@DeclareRobustCommand\@undefined
671 <latexrelease>\let\@if@newcommand\@undefined
672 <latexrelease>\let\copy@newcommand\@undefined
673 <latexrelease>\let\show@newcommand\@undefined
674 %
675 <latexrelease>\let\copy@kernel@robust@command\@undefined
676 <latexrelease>\let\show@kernel@robust@command\@undefined
677 <latexrelease>\let\show@newcommand@aux\@undefined
678 <latexrelease>\EndIncludeInRelease
679 <*2ekernel>
```

(End of definition for `\@if@newcommand` and others.)

1.5.5 Showing environments

`\ShowEnvironment`

```
680 </2ekernel>
681 <latexrelease>\IncludeInRelease{2023-06-01}{\ShowEnvironment}
682 <latexrelease> {Add \ShowEnvironment}%
683 <*2ekernel | latexrelease>
```

`\ShowEnvironment` is quite similar to `\ShowCommand`. We will pass the environment `(env)` around as the macro `\env`, because `\robust@command@act` expects a single token.

```
684 \def\ShowEnvironment#1{%
685   \expandafter\@show@environment\csname #1\endcsname}
686 \long\def\@show@environment#1{%
687   \robust@command@act}
```

```

688     \@showenvironmentlisthook#1%
689     \@show@normalenv#1}

```

This is similar to `\@showcommandlisthook`, but uses the dedicated versions for environments.

```

690 \def\@showenvironmentlisthook{%
691   {\@if@DeclareRobustCommand \@show@DeclareRobustCommand@env}%
692   {\@if@newcommand \@show@newcommand@env}}

```

These are similar to the command versions below, except they say “environment” and call `\@show@environment@end` to print the `\end` part.

```

693 \long\def\@show@newcommand@env#1{%
694   \@show@environment@begin#1%
695   \expandafter\@show@newcommand@aux
696   \csname\@backslashchar\@expl@cs@to@str@@N#1\expandafter\endcsname
697   \expandafter{\#1}\@show@typeout
698   \@show@environment@end#1}
699 \long\def\@show@DeclareRobustCommand@env#1{%
700   \@show@environment@begin#1%
701   \begingroup
702   \let\@show@tokens\@show@typeout
703   \let\@show@macro\@show@nonstop
704   \expandafter\show@kernel@robust@command
705   \csname\@expl@cs@to@str@@N#1 \endcsname
706   \endgroup
707   \@show@environment@end#1}
708 \long\def\@show@environment@begin#1{%
709   \typeout{> \string\begin{\@expl@cs@to@str@@N#1}=environment :}%
710   \typeout{\@expl@cs@argument@spec@@N#1->%
711           \@expl@cs@replacement@spec@@N#1.^~J}}

```

A “normal” environment is straightforward. `\@show@environment@end` needs to check if the `\end` part is defined and show it accordingly, otherwise the output would show gibberish.

```

712 \long\def\@show@normalenv#1{%
713   \@show@environment@begin#1%
714   \@show@environment@end#1}
715 \long\def\@show@environment@end#1{%
716   \expandafter\@show@environment@end@aux
717   \csname end\@expl@cs@to@str@@N#1\endcsname#1}
718 \long\def\@show@environment@end@aux#1#2{%
719   \@show@tokens{\string\end{\@expl@cs@to@str@@N#2}%
720   \ifx\relax#1=undefined%
721   \else:^~J\@expl@cs@argument@spec@@N#1->%
722   \@expl@cs@replacement@spec@@N#1%
723   \fi}}

```

And here some auxiliaries:

```

\@show@nonstop
\@show@typeout

```

`\@show@nonstop` same output as `\show`, but doesn’t stop for interaction;

`\@show@typeout` same output as `\showtokens`, but doesn’t stop for interaction.

```

724 \def\@show@nonstop#1{%
725   \typeout{> \string#1=\@expl@cs@prefix@spec@@N#1macro:^^J%
726   \@expl@cs@argument@spec@@N#1->\@expl@cs@replacement@spec@@N#1.}}
727 \def\@show@typeout#1{\typeout{> #1.^^J}}

```

Now the rollback code.

```

728 {/2ekernel | latexrelease}
729 <latexrelease>\EndIncludeInRelease
730 <latexrelease>\IncludeInRelease{0000-00-00}{\ShowEnvironment}
731 <latexrelease> {Undefine \ShowEnvironment}%
732 <latexrelease>\let\ShowEnvironment\undefined
733 <latexrelease>\EndIncludeInRelease
734 {*2ekernel}

```

(End of definition for \ShowEnvironment 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,

```

735 {/2ekernel}
736 <latexrelease>\IncludeInRelease{2018-04-01}{\@ifundefined}
737 <latexrelease>{Leave commands undefined in \@ifundefined}%
738 {*2ekernel | latexrelease}

```

Version using \ifcsname to avoid defining undefined tokens to \relax. Defined here to simplify using unmatched \fi.

```

739 \def\@ifundefined#1{%
740   \ifcsname#1\endcsname\@ifundefined@d@i\else\@ifundefined@d@ii\fi{#1}}
741 \long\def\@ifundefined@d@i#1\fi#2{#1}
742   \expandafter\ifx\csname #2\endcsname\relax
743     \@ifundefined@d@ii
744   \fi
745   \@secondoftwo}
746 \long\def\@ifundefined@d@ii\fi#1#2#3{\fi #2}

```

Now test of engine.

```
747 \ifx\numexpr\undefined
```

Classic version (should not be needed as etex is assumed).

```

748 \def\@ifundefined#1{%
749   \expandafter\ifx\csname#1\endcsname\relax
750     \expandafter\@firstoftwo
751   \else
752     \expandafter\@secondoftwo
753   \fi}
754 \else\ifx\directlua\undefined

```

Use the \ifcsname defined above.

```
755 \else
```

Optimised version for LuaTeX, using \lastnamedcs

```
756 \def\@ifundefined#1{%
757   \ifcsname#1\endcsname
758     \expandafter\ifx\lastnamedcs\relax\else\@ifundefined@#1\fi
759   \fi
760   \@firstoftwo}
761 \long\def\@ifundefined@#1#2#3#4#5{#1#2#5}
762 \fi
763 \fi
764 </2ekernel | latexrelease>
765 <latexrelease>\EndIncludeInRelease
766 <latexrelease>\IncludeInRelease{0000-00-00}{\@ifundefined}
767 <latexrelease>\{Leave commands undefined in \@ifundefined\}%
768 <latexrelease>\def\@ifundefined#1{%
769   <latexrelease> \expandafter\ifx\csname#1\endcsname\relax
770   <latexrelease> \expandafter\@firstoftwo
771   <latexrelease> \else
772   <latexrelease> \expandafter\@secondoftwo
773   <latexrelease> \fi}
774 <latexrelease>\EndIncludeInRelease
775 <*2ekernel>
```

(End of definition for \@ifundefined.)

\@qend The following define \@qend and \@qrelax to be the strings ‘end’ and ‘relax’ with the characters \catcode 12.

```
776 \edef\@qend{\expandafter\@cdr\string\end\@nil}
777 \edef\@qrelax{\expandafter\@cdr\string\relax\@nil}
```

(End of 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.

```
778 \long\def\@ifnextchar#1#2#3{%
779   \let\reserved@d=#1%
780   \def\reserved@a{#2}%
781   \def\reserved@b{#3}%
782   \futurelet\@let@token\@ifnch}
```

(End of 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.

```
783 \let\kernel@ifnextchar\@ifnextchar
```

(End of 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.

```

784 \def\@ifnch{%
785   \ifx\@let@token\@sptoken
786     \let\reserved@c\@xifnch
787   \else
788     \ifx\@let@token\reserved@d
789       \let\reserved@c\reserved@a
790     \else
791       \let\reserved@c\reserved@b
792     \fi
793   \fi
794 }
```

(End of 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.

```
795 \def\:{\let\@sptoken= } \: % this makes \@sptoken a space token
```

(End of definition for \@sptoken.)

\@xifnch In the following definition of \@xifnch, \: is again used to get a space token as delimiter into the definition.

```
796 \def\:{\@xifnch} \expandafter\def\:\ {\futurelet\@let@token\@ifnch}
```

(End of 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.

```
797 \def\@ifstar#1{\@ifnextchar *{\@firstoftwo{#1}}{}}
```

(End of definition for \@ifstar.)

```

\@dblarg
\@xdblarg
798 \long\def\@dblarg#1{\kernel@ifnextchar[{#1}]{\@dblarg{#1}}}
799 \long\def\@xdblarg#1#2{#1[#2]}{#2}}
```

(End of 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.

```

800 \def\@sanitize{\@makeother\ \ @makeother\\\ @makeother\$ @makeother\&%
801 \ @makeother\# @makeother\^ @makeother\_ @makeother\% @makeother\~}
```

(End of definition for \@sanitize.)

`\@onellevel@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.

```
802 \def \@onellevel@sanitize #1{%
803   \edef #1{\expandafter\strip@prefix
804           \meaning #1}%
805 }
```

(End of definition for `\@onellevel@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.

```
806 </2ekernel>
807 <latexrelease>\IncludeInRelease{2020/10/01}{\string@makeletter}
808 <latexrelease> {Add \string@makeletter}%
809 <*2ekernel | latexrelease>
810 \def\string@makeletter#1{%
811   \@expl@str@map@function@@NN#1\@string@makeletter}
812 \def\@string@makeletter#1{%
813   \char@if@alph{#1}%
814   {\@expl@char@generate@nn{'#1}{11}}%
815   {#1}}
816 \def\char@if@alph#1{%
817   \ifnum0\ifnum`#1<`A 1\fi\ifnum`#1>`z 1\fi
818   \if\ifnum`#1>`Z @\fi\ifnum`#1<`a @\fi01\fi>0
819   \expandafter\@secondoftwo
820 \else
821   \expandafter\@firstoftwo
822 \fi}
823 </2ekernel | latexrelease>
824 <latexrelease>\EndIncludeInRelease
825 %
826 <latexrelease>\IncludeInRelease{0000/00/00}{\string@makeletter}
827 <latexrelease> {Undefined \string@makeletter}%
828 <latexrelease>\let\string@makeletter\@undefined
829 <latexrelease>\let\@string@makeletter\@undefined
830 <latexrelease>\let\char@if@alph\@undefined
831 <latexrelease>\EndIncludeInRelease
832 <*2ekernel>
```

(End of definition for `\string@makeletter`, `\@string@makeletter`, and `\char@if@alph`.)

`\makeatletter` Make internal control sequences accessible or inaccessible.

```
833 \DeclareRobustCommand\makeatletter{\catcode`\@11\relax}
834 \DeclareRobustCommand\makeatother{\catcode`\@12\relax}
```

(End of definition for `\makeatletter` and `\makeatother`.)

2 Discretionary Hyphenation

```
\-
\@dischyp 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.
```

L^AT_EX, from L^AT_EX2.09 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 T_EX definition is saved as \@@hyph just in case anyone
needs it.

L^AT_EX 2_E, 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 L^AT_EX definition
accessible via `latexrelease` as usual.

In LuaL^AT_EX 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 LuaL^AT_EX compatible with language specific hyphenation
characters.

Temporary definition of \@latex@info, final definition is later.

```
835 \def\@latex@info#1{%
836   \ifx\@kernel\@latexrelease\else\@latexrelease\fi
837   {  
838     \ifx\@kernel\@latexrelease\else\@latexrelease\fi
839     \ifx\directlua\undefined
840       \ DeclareRobustCommand{\-}{%
841         \discretionary{-}{ }{ }\@ifnum\hyphenchar>=0
842           \def\@hyphenchar{\hyphenchar}
843         \else
844           \def\@hyphenchar{\hyphenchar}
845         \fi
846         }{ }%
847       }{ }%
848     }
849   \else
850     \let\-\@hyph
851   \fi
852   \ifx\@kernel\@latexrelease\else\@latexrelease\fi
853   {  
854     \ifx\@kernel\@latexrelease\else\@latexrelease\fi
855     \ifx\@kernel\@latexrelease\else\@latexrelease\fi
856     \ifx\@kernel\@latexrelease\else\@latexrelease\fi
857     \ifx\@kernel\@latexrelease\else\@latexrelease\fi
858     \ifx\@kernel\@latexrelease\else\@latexrelease\fi
859     \ifx\@kernel\@latexrelease\else\@latexrelease\fi
860   }
```

```

860 〈\latexrelease〉          \hyphenchar\font
861 〈\latexrelease〉          \fi
862 〈\latexrelease〉          }{}{%
863 〈\latexrelease〉
864 〈\latexrelease〉\EndIncludeInRelease
865 〈\latexrelease〉\IncludeInRelease{0000/00/00}{\-}{Use \hyphenchar in \-}%
866 〈\latexrelease〉\def\-\{\discretionary{-}{}{}\}
867 〈\latexrelease〉\EndIncludeInRelease

868 〈*2ekernel | \latexrelease〉
869 〈\let\@dischyp=\-
870 〈/2ekernel | \latexrelease〉
871 〈*2ekernel〉

(End of definition for \- and \@dischyp.)
Delayed from ltvers.dtx
872 \newif\if@includeinrelease
873 \if@includeinreleasefalse
Delayed from ltplain.dtx
874 〈/2ekernel〉
875 〈*2ekernel | \latexrelease〉
876 〈\latexrelease〉\IncludeInRelease{2019/10/01}%
877 〈\latexrelease〉          {\allowbreak}{Make various commands robust}%
878 \MakeRobust\allowbreak
879 \MakeRobust\bigbreak
880 \MakeRobust\break
881 \MakeRobust\dotfill
882 \MakeRobust\frenchspacing
883 \MakeRobust\goodbreak
884 \MakeRobust\hrulefill
885 \MakeRobust\medbreak
886 \MakeRobust\nobreak
887 \MakeRobust\nonfrenchspacing
888 \MakeRobust\obeylelines
889 \MakeRobust\obeyspaces
890 \MakeRobust\slash
891 \MakeRobust\smallbreak
892 \MakeRobust\strut
893 \MakeRobust\underbar
894 〈/2ekernel | \latexrelease〉
895 〈\latexrelease〉\EndIncludeInRelease
896 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
897 〈\latexrelease〉          {\allowbreak}{Make various commands robust}%
898 〈\latexrelease〉
899 〈\latexrelease〉\kernel@make@fragile\allowbreak
900 〈\latexrelease〉\kernel@make@fragile\bigbreak
901 〈\latexrelease〉\kernel@make@fragile\break
902 〈\latexrelease〉\kernel@make@fragile\dotfill
903 〈\latexrelease〉\kernel@make@fragile\frenchspacing
904 〈\latexrelease〉\kernel@make@fragile\goodbreak
905 〈\latexrelease〉\kernel@make@fragile\hrulefill
906 〈\latexrelease〉\kernel@make@fragile\medbreak
907 〈\latexrelease〉\kernel@make@fragile\nobreak
908 〈\latexrelease〉\kernel@make@fragile\nonfrenchspacing

```

```
909 〈\latexrelease〉\kernel@make@fragile\obeylines
910 〈\latexrelease〉\kernel@make@fragile\obeyspaces
911 〈\latexrelease〉\kernel@make@fragile\slash
912 〈\latexrelease〉\kernel@make@fragile\smallbreak
913 〈\latexrelease〉\kernel@make@fragile\strut
914 〈\latexrelease〉\kernel@make@fragile\underbar
915 〈\latexrelease〉
916 〈\latexrelease〉\EndIncludeInRelease
917 〈*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.

```
918 \long\def\g@addto@macro#1#2{%
919   \begingroup
920     \toks@\expandafter{\#1#2}%
921     \xdef#1{\the\toks@}%
922   \endgroup}
```

(*End of definition for \g@addto@macro.*)

```
923 〈/2ekernel〉
```

File g

ltcmd.dtx

1 Creating document commands

Document commands should be created using the tools provided by this module: `\NewDocumentCommand`, etc., in almost all cases. This allows clean separation of document-level syntax from code-level interfaces. Users have a need to create new document commands, and as such a significant amount of documentation for `ltcmd` is provided as part of `usrguide3`. Here, additional material aimed at programmers is provided

```
1 <@@=cmd>
2 <*2ekernel>
3 \message{document commands,}
4 </2ekernel>
```

`ltcmd` code contains an `\^@` character, which usually has catcode 15, so `\IncludeInRelease` will break when this code is being skipped, so we'll save the catcode of `\^@` to restore later:

```
5 <*2ekernel | latexrelease>
6 <| latexrelease>\edef\@latexrelease@catcode@null{\the\catcode'\^@ }
7 <| latexrelease>\catcode'\^@=12
8 \ExplSyntaxOn
9 <| latexrelease>\NewModuleRelease{2020/10/01}{ltcmd}
10 <| latexrelease> {Document~command~parser} %
```

1.1 Variables and constants

`\l__cmd_arg_spec_tl` Holds the argument specification after normalization of shorthands.

```
11 \tl_new:N \l__cmd_arg_spec_tl
```

`\l__cmd_args_tl` Token list variable for grabbed arguments.

```
12 \tl_new:N \l__cmd_args_tl
```

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

`\l__cmd_args_ii_tl`

```
13 \tl_new:N \l__cmd_args_i_tl
14 \tl_new:N \l__cmd_args_ii_tl
```

`\l__cmd_current_arg_int` The number of the current argument being set up: this is used to make sure there are at most 9 arguments, then for creating the expandable auxiliary functions and knowing how many arguments the code function should take.

```
15 \int_new:N \l__cmd_current_arg_int
```

\l__cmd_defaults_bool The boolean indicates whether there are any argument with default value other than -*NoValue*-; the token list holds the code to determine these default values in terms of other arguments.

```
16 \bool_new:N \l__cmd_defaults_bool  
17 \tl_new:N \l__cmd_defaults_tl
```

\l__cmd_environment_bool Generating environments uses the same mechanism as generating functions. However, full processing of arguments is always needed for environments, and so the function-generating code needs to know this. This variable is also used at run time to give correct error messages.

```
18 \bool_new:N \l__cmd_environment_bool
```

\l__cmd_environment_str Name of the environment, used at definition time and at run time.

```
19 \str_new:N \l__cmd_environment_str
```

\l__cmd_expandable_bool Used to indicate if an expandable command is being generated, as this affects both the acceptable argument types and how they are implemented.

```
20 \bool_new:N \l__cmd_expandable_bool
```

\l__cmd_expandable_aux_name_tl

Used to create pretty-printing names for the auxiliaries: although the immediate definition does not vary, the full expansion does and so it does not count as a constant.

```
21 \tl_new:N \l__cmd_expandable_aux_name_tl  
22 \tl_set:Nn \l__cmd_expandable_aux_name_tl  
23 {  
24     \l__cmd_function_tl \c_space_tl  
25     ( arg~ \int_use:N \l__cmd_current_arg_int )  
26 }
```

\g__cmd_grabber_int Used (in exceptional cases) to get unique names for grabbers used by expandable commands.

```
27 \int_new:N \g__cmd_grabber_int
```

\l__cmd_fn_tl For passing the pre-formed name of the auxiliary to be used as the parsing function.

```
28 \tl_new:N \l__cmd_fn_tl
```

\l__cmd_fn_code_tl For passing the pre-formed name of the auxiliary that contains the actual code.

```
29 \tl_new:N \l__cmd_fn_code_tl
```

\l__cmd_function_tl Holds the control sequence name of the function currently being defined: used to avoid passing this as an argument and to avoid repeated use of \cs_to_str:N.

³⁰ \tl_new:N \l__cmd_function_tl

\l__cmd_grab_expandably_bool

When defining a non-expandable command, indicates whether the arguments can all safely be grabbed by expandable grabbers. This is to support abuses of *xparse* that use protected functions inside csname constructions.

³¹ \bool_new:N \l__cmd_grab_expandably_bool

\l__cmd_obey_spaces_bool For trailing optionals.

³² \bool_new:N \l__cmd_obey_spaces_bool

\l__cmd_last_delimiters_tl Holds the delimiters (first tokens) of all optional arguments since the previous mandatory argument, to warn about cases where it would be impossible to omit optional arguments completely because the following mandatory argument has the same delimiter as one of the optional arguments.

³³ \tl_new:N \l__cmd_last_delimiters_tl

\l__cmd_long_bool Used to indicate that an argument is long, on a per-argument basis.

³⁴ \bool_new:N \l__cmd_long_bool

\l__cmd_suppress_strip_bool

Used to indicate that an a pair of braces should not be stripped from an optional argument.

³⁵ \bool_new:N \l__cmd_suppress_strip_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.

³⁶ \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).

³⁷ \bool_new:N \l__cmd_prefixed_bool

`\l__cmd_process_all_t1` When preparing the signature, the processors that will be applied to a given argument are collected in `\l__cmd_process_one_t1`, while `\l__cmd_process_all_t1` contains processors for all arguments. The boolean indicates whether there are any processors (to bypass the whole endeavour otherwise).

```
38 \tl_new:N \l__cmd_process_all_t1
39 \tl_new:N \l__cmd_process_one_t1
40 \bool_new:N \l__cmd_process_some_bool
```

`\l__cmd_saved_args_t1` Stores `\l__cmd_args_t1` to deal with space-trimming of b-type arguments.

```
41 \tl_new:N \l__cmd_saved_args_t1
```

`\l__cmd_signature_t1` 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.

```
42 \tl_new:N \l__cmd_signature_t1
```

`\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.

```
43 \bool_new:N \l__cmd_some_obey_spaces_bool
44 \bool_new:N \l__cmd_some_long_bool
45 \bool_new:N \l__cmd_some_short_bool
```

`\q__cmd_recursion_tail`
`_cmd_if_recursion_stop` Quarks and functions for internal processing.

```
46 \quark_new:N \q__cmd_recursion_tail
47 \quark_new:N \q__cmd_recursion_stop
48 \_kernel_quark_new_test:N \_cmd_if_recursion_tail_stop_do:Nn
```

(End of definition for `_cmd_if_recursion_tail_stop_do:Nn` and
`_cmd_use_i_delimit_by_q_recursion_stop:nw`.)

```
\l__cmd_tmp_prop
\l_\cmdtmpapt\ Scratch space.
\l__cmd_tmpb_tl
```

49 \prop_new:N \l__cmd_tmp_prop
50 \tl_new:N \l__cmd_tmpa_tl
51 \tl_new:N \l__cmd_tmpb_tl
52 \cs_new_eq:NN __cmd_tmp:w ?

(End of definition for __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.

```
53 \msg_redirect_module:nnn { cmd } { info } { none }
Also add cmd to the LaTeX messages.
54 \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.

```
55 \cs_new_protected:Npn \__cmd_declare_cmd:Nnn
56 {
57     \bool_set_false:N \l__cmd_expandable_bool
58     \__cmd_declare_cmd_aux:Nnn
59 }
60 \cs_new_protected:Npn \__cmd_declare_expandable_cmd:Nnn
61 {
62     \bool_set_true:N \l__cmd_expandable_bool
63     \__cmd_declare_cmd_aux:Nnn
64 }
```

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.

```
65 \cs_new_protected:Npn \__cmd_declare_cmd_aux:Nnn #1#2#3
66 {
67     \cs_if_exist:NTF #1
68     {
69         \msg_info:nnxx { cmd } { redefine }
70         { \token_to_str:N #1 } { \tl_to_str:n {#2} }
71     }
72     {
73         \bool_lazy_or:nnT
74         { \cs_if_exist_p:c { \cs_to_str:N #1 ~ code } }
75         { \cs_if_exist_p:c { \cs_to_str:N #1 ~ defaults } }
76         {
77             \msg_warning:nnx { cmd } { unsupported-let }
78             { \token_to_str:N #1 }
79         }
80         \msg_info:nnxx { cmd } { define-command }
81         { \token_to_str:N #1 } { \tl_to_str:n {#2} }
82     }
83     \bool_set_false:N \l__cmd_environment_bool
84     \__cmd_declare_cmd_internal:Nnnn #1 {#2} {#3} { }
```

```
85     }
```

At definition time, the variable `\l__cmd_fn_t1` 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.

```
86 \cs_new_protected:Npn \__cmd_declare_cmd_internal:Nnnn #1#2#3#4
87 {
88     \tl_set:Nx \l__cmd_function_t1 { \cs_to_str:N #1 }
89     \tl_set:Nx \l__cmd_fn_t1
90         { \exp_not:c { \l__cmd_function_t1 \c_space_t1 } }
91     \__cmd_normalize_arg_spec:n {#2}
92     \exp_args:No \__cmd_prepare_signature:n \l__cmd_arg_spec_t1
93     \__cmd_declare_cmd_code:Nnn #1 {#2} {#3}
94     #4
95     \__cmd_break_point:n {#2}
96 }
```

(End of definition for `__cmd_declare_cmd:Nnn` and others.)

`__cmd_break_point:n` A marker used to escape from creating a definition if necessary.

```
97 \cs_new_eq:NN \__cmd_break_point:n \use_none:n
```

(End of definition for `__cmd_break_point:n`.)

`__cmd_declare_cmd_code:Nnn` The appropriate auxiliary is called.

```
98 \cs_new_protected:Npn \__cmd_declare_cmd_code:Nnn
99 {
100     \bool_if:NTF \l__cmd_grab_expandably_bool
101         { \__cmd_declare_cmd_code_expandable:Nnn }
102         { \__cmd_declare_cmd_code_aux:Nnn }
103 }
```

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.

```
104 \cs_new_protected:Npn \__cmd_declare_cmd_code_aux:Nnn #1#2#3
105 {
106     \cs_generate_from_arg_count:cNnn
107         { \l__cmd_function_t1 \c_space_t1 code }
108         \cs_set_protected:Npn \l__cmd_current_arg_int {#3}
109     \cs_set_protected_nopar:Npx #1
110     {
111         \bool_if:NTF \l__cmd_environment_bool
112             {
113                 \__cmd_start_env:nnnnn { \exp_not:n {#2} }
114                 { \l__cmd_environment_str }
115             }
116             {
117                 \__cmd_start:nNnnnn { \exp_not:n {#2} }
118                 \exp_not:c { \l__cmd_function_t1 \c_space_t1 }
119                 \exp_not:c { \l__cmd_function_t1 \c_space_t1 code }
120             }
121             { \exp_not:o \l__cmd_signature_t1 }
122             {
123                 \bool_if:NT \l__cmd_defaults_bool
```

```

124          { \exp_not:o \l__cmd_defaults_tl }
125      }
126      {
127          \bool_if:NT \l__cmd_process_some_bool
128          { \exp_not:o \l__cmd_process_all_tl }
129      }
130  }
131 }

```

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 csname. 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 (the only) grabber is defined correspondingly to be long/short. Otherwise two grabbers are defined, one long, one short.

```

132 \cs_new_protected:Npn \__cmd_declare_cmd_code_expandable:Nnn #1#2#3
133 {
134     \exp_args:Ncc \cs_generate_from_arg_count>NNnn
135     { \l__cmd_function_tl \c_space_tl code }
136     { cs_set \bool_if:NF \l__cmd_expandable_bool { _protected } :Npn }
137     \l__cmd_current_arg_int {#3}
138     \bool_if:NT \l__cmd_defaults_bool
139     {
140         \use:x
141         {
142             \cs_generate_from_arg_count:cNnn
143             { \l__cmd_function_tl \c_space_tl defaults }
144             \cs_set:Npn \l__cmd_current_arg_int
145             { \exp_not:o \l__cmd_defaults_tl }
146         }
147     }
148     \bool_if:NTF \l__cmd_expandable_bool
149     { \cs_set_nopar:Npx } { \cs_set_protected_nopar:Npx } #1
150     {
151         \exp_not:N \__cmd_start_expandable:nNNNNn
152         { \exp_not:n {#2} }
153         \exp_not:c { \l__cmd_function_tl \c_space_tl }
154         \exp_not:c
155         {
156             \l__cmd_function_tl \c_space_tl
157             \bool_if:NT \l__cmd_some_short_bool
158             { \bool_if:NT \l__cmd_some_long_bool { \c_space_tl } }
159         }
160         \exp_not:c { \l__cmd_function_tl \c_space_tl code }
161         \bool_if:NTF \l__cmd_defaults_bool
162         { \exp_not:c { \l__cmd_function_tl \c_space_tl defaults } }
163         { ? }
164         { \exp_not:o \l__cmd_signature_tl }
165     }
166     \bool_if:NTF \l__cmd_some_long_bool

```

```

167 {
168   \bool_if:NT \l__cmd_some_short_bool
169   {
170     \cs_set_nopar:cpx { \l__cmd_function_tl \c_space_tl \c_space_tl }
171     ##1##2 { ##1 {##2} }
172   }
173   \cs_set:cpx
174 }
175 { \cs_set_nopar:cpx
176   { \l__cmd_function_tl \c_space_tl } ##1##2 { ##1 {##2} }
177 }

```

(End of definition for __cmd_declare_cmd_code:Nnn, __cmd_declare_cmd_code_aux:Nnn, and __cmd_declare_cmd_code_expandable:Nnn.)

__cmd_declare_env:nnnn
__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.

```

178 \cs_new_protected:Npn \__cmd_declare_env:nnnn #1#2
179 {
180   \str_set:Nx \l__cmd_environment_str {#1}
181   \str_set:Nx \l__cmd_environment_str
182   { \tl_trim_spaces:o { \l__cmd_environment_str } }
183   \cs_if_exist:cTF { \l__cmd_environment_str }
184   {
185     \msg_info:nnxx { cmd } { redefine-env }
186     { \l__cmd_environment_str } { \tl_to_str:n {#2} }
187   }
188   {
189     \msg_info:nnxx { cmd } { define-env }
190     { \l__cmd_environment_str } { \tl_to_str:n {#2} }
191   }
192   \bool_set_false:N \l__cmd_expandable_bool
193   \bool_set_true:N \l__cmd_environment_bool
194   \exp_args:NV \__cmd_declare_env_internal:nnnn
195   \l__cmd_environment_str {#2}
196 }

```

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.

```

197 \cs_new_protected:Npn \__cmd_declare_env_internal:nnnn #1#2#3#4
198 {
199   \exp_args:Nc \__cmd_declare_cmd_internal:Nnnn { environment~ #1 } {#2}
200   {#3}
201   {
202     \cs_set_nopar:cpx { environment~ #1 ~end }
203     { \exp_not:c { environment~ #1 ~end~aux } }

```

```

204     \cs_generate_from_arg_count:cNnn
205     { environment~ #1 ~end~aux~ } \cs_set:Npn
206     \l__cmd_current_arg_int {#4}
207     \cs_set_eq:cc {#1} { environment~ #1 }
208     \cs_set_eq:cc { end #1 } { environment~ #1 ~end }
209   }
210 }
211 \cs_new_protected:Npn \__cmd_set_environment_end:n #1
212 {
213   \cs_set_nopar:cp { environment~ #1 ~end~aux }
214   {
215     \exp_not:c { environment~ #1 ~end~aux~ }
216     \exp_not:o \l__cmd_args_tl
217   }
218 }

```

(End of definition for `__cmd_declare_env:nnnn` and `__cmd_declare_env_internal:nnnn`.)

1.3 Structure of `xparse` commands

```
\__cmd_start_env:nnnnn
\__cmd_start:nNNnnn
```

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`).

```

219 \cs_new_protected:Npn \__cmd_start_env:nnnnn #1#2
220 {
221   \conditionally@traceoff
222   \group_align_safe_begin:
223   \str_set:Nn \l__cmd_environment_str {#2}
224   \bool_set_true:N \l__cmd_environment_bool
225   \__cmd_start_aux:ccnnnn
226   { environment~ \l__cmd_environment_str \c_space_tl }
227   { environment~ \l__cmd_environment_str \c_space_tl code }
228   {#1}
229 }
230 \cs_new_protected:Npx \__cmd_start:nNNnnn #1#2#3
231 {
232   \exp_not:c { xparse~function~is~not~expandable }
233   \exp_not:N \conditionally@traceoff
234   \exp_not:N \group_align_safe_begin:
235   \exp_not:n { \bool_set_false:N \l__cmd_environment_bool }
236   \exp_not:N \__cmd_start_aux:NNnnnn
237   #2 #3 {#1}
238 }
```

(End of definition for `__cmd_start_env:nnnnn` and `__cmd_start:nNNnnn`.)

```
\__cmd_start_aux:NNnnnn
\__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.

```

239 \cs_new_protected:Npn \__cmd_start_aux:NNnnnn #1#2#3#4#5#6
240 {
241     \tl_clear:N \l__cmd_args_tl
242     \tl_set:Nn \l__cmd_fn_tl {#1}
243     \tl_set:Nn \l__cmd_fn_code_tl {#2}
244     \tl_set:Nn \l__cmd_defaults_tl {#5}
245     \tl_set:Nn \l__cmd_process_all_tl {#6}
246     #4
247     \__cmd_run_code:
248 }
249 \cs_generate_variant:Nn \__cmd_start_aux:NNnnnn { cc }

(End of 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.

```

250 \cs_new_protected:Npn \__cmd_run_code:
251 {
252     \tl_if_empty:NF \l__cmd_defaults_tl { \__cmd_defaults: }
253     \tl_if_empty:NF \l__cmd_process_all_tl { \__cmd_args_process: }
254     \bool_if:NT \l__cmd_environment_bool
255         { \exp_args:No \__cmd_set_environment_end:n \l__cmd_environment_str }
256     \group_align_safe_end:
257     \conditionally@traceon
258     \exp_after:wN \l__cmd_fn_code_tl \l__cmd_args_tl
259 }

```

(End of definition for __cmd_run_code:.)

__cmd_defaults:
__cmd_defaults_def:
__cmd_defaults_def:nn
__cmd_defaults_def:nnn
__cmd_defaults_aux:
__cmd_defaults_error:w

First construct __cmd_tmp:w (see below) that will receive the arguments found so far and determine default values for any missing argument. Then call it repeatedly until the set of arguments stabilizes. Since that could lead to an infinite loop we only call it up to nine times, the maximal number needed for stabilization if there is a chain of arguments that depend on each other. If that fails to stabilize raise an error.

```

260 \cs_new_protected:Npn \__cmd_defaults:
261 {
262     \__cmd_defaults_def:
263     \tl_set_eq:NN \l__cmd_args_i_tl \l__cmd_args_tl
264     \__cmd_defaults_aux: \__cmd_defaults_aux: \__cmd_defaults_aux:
265     \__cmd_defaults_aux: \__cmd_defaults_aux: \__cmd_defaults_aux:
266     \__cmd_defaults_aux: \__cmd_defaults_aux: \__cmd_defaults_aux:
267     \__cmd_defaults_error:w
268     \q_recursion_stop
269     \tl_set_eq:NN \l__cmd_args_tl \l__cmd_args_i_tl
270 }
271 \cs_new_protected:Npn \__cmd_defaults_aux:
272 {
273     \tl_set:Nx \l__cmd_args_ii_tl
274         { \exp_after:wN \__cmd_tmp:w \l__cmd_args_i_tl }
275     \tl_if_eq:NNT \l__cmd_args_ii_tl \l__cmd_args_i_tl
276         { \use_none_delimit_by_q_recursion_stop:w }
277     \tl_set_eq:NN \l__cmd_args_i_tl \l__cmd_args_ii_tl
278 }

```

```

279 \cs_new_protected:Npn \__cmd_defaults_error:w \q_recursion_stop
280   {
281     \msg_error:nnx { cmd } { default-loop }
282     { \__cmd_environment_or_command: }
283   }

```

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.

```

284 \cs_new_protected:Npn \__cmd_defaults_def:
285   {
286     \tl_clear:N \l__cmd_tmpa_tl
287     \int_zero:N \l__cmd_current_arg_int
288     \__cmd_tl_mapthread_function:NNN \l__cmd_args_tl \l__cmd_defaults_tl
289     \__cmd_defaults_def:nn
290     \cs_generate_from_arg_count:NNno \__cmd_tmp:w \cs_set:Npn
291       \l__cmd_current_arg_int \l__cmd_tmpa_tl
292   }
293 \cs_generate_variant:Nn \cs_generate_from_arg_count:NNnn { NNno }
294 \cs_new_protected:Npn \__cmd_defaults_def:nn
295   {
296     \int_incr:N \l__cmd_current_arg_int
297     \exp_args:NV \__cmd_defaults_def:nnn \l__cmd_current_arg_int
298   }
299 \cs_new_protected:Npn \__cmd_defaults_def:nnn #1#2#3
300   {
301     \tl_put_right:Nx \l__cmd_tmpa_tl
302     {
303       {
304         \exp_not:N \exp_not:n
305         {
306           \tl_if_novalue:nTF {#2}
307             { \exp_not:o {#3} }
308             { \exp_not:n { ## #1 } }
309         }
310       }
311     }
312   }

```

(End of definition for `__cmd_defaults:` and others.)

`__cmd_args_process:` 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`.

```

313 \cs_new_protected:Npn \__cmd_args_process:
314   {
315     \tl_clear:N \l__cmd_args_ii_tl
316     \__cmd_tl_mapthread_function:NNN
317       \l__cmd_args_tl
318       \l__cmd_process_all_tl
319       \__cmd_args_process_loop:nn

```

```

320      \tl_set_eq:NN \l__cmd_args_tl \l__cmd_args_ii_tl
321  }
322 \cs_new_protected:Npn \__cmd_args_process_loop:nn #1#2
323 {
324     \tl_set:Nn \ProcessedArgument {#1}
325     \tl_if_no_value:nF {#1}
326     { \tl_map_function:nN {#2} \__cmd_args_process_aux:n }
327     \tl_put_right:No \l__cmd_args_ii_tl
328     { \exp_after:wN { \ProcessedArgument } }
329 }
330 \cs_new_protected:Npn \__cmd_args_process_aux:n #1
331 {
332     \cs_generate_from_arg_count:NNnn \__cmd_tmp:w \cs_set:Npn
333     { \tl_count:N \l__cmd_args_tl } {#1}
334     \exp_args:NNNo \exp_after:wN \__cmd_tmp:w \l__cmd_args_tl
335     { \ProcessedArgument }
336 }

```

(End of definition for `__cmd_args_process:`, `__cmd_args_process_loop:nn`, and `__cmd_args_process_aux:n`.)

`__cmd_start_expandable:nNNNNn`

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.

```

337 \cs_new:Npn \__cmd_start_expandable:nNNNNn #1#2#3#4#5#6
338 {
339     \group_align_safe_begin:
340     #6 \__cmd_end_expandable:NNw #5 #4 \q__cmd #2#3
341 }

```

(End of definition for `__cmd_start_expandable:nNNNNn`.)

`__cmd_end_expandable:NNw`
`__cmd_end_expandable_aux:w`
`__cmd_end_expandable_defaults:nNNNN`
`__cmd_end_expandable_defaults:nnw`
`__cmd_end_expandable_defaults:nw`

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:nNNNN`).

```

342 \cs_new:Npn \__cmd_end_expandable:NNw #1#2
343   { \__cmd_end_expandable_aux:w #1#2 \prg_do_nothing: }
344 \cs_new:Npn \__cmd_end_expandable_aux:w #1#2#3 \q__cmd
345   { \exp_args:No \__cmd_end_expandable_aux:nNNNN {#3} #1 #2 }
346 \cs_new:Npn \__cmd_end_expandable_aux:nNNNN #1#2#3#4#5
347 {
348     \token_if_eq_charcode:NNT ? #2 { \exp_after:wN \use_iv:nnnn }
349     \__cmd_end_expandable_defaults:nNNNN {#1} { } {#1} #2#3
350     { } { } { } { } { } { } { } { } { } { }

```

```

351      {
352          \msg_expandable_error:nnf { cmd } { default-loop }
353          { \exp_args:Nf \tl_trim_spaces:n { \token_to_str:N #4 } }
354          \use_iv:nnnn
355      }
356      \q_stop
357  }
358 \cs_new:Npn \__cmd_end_expandable_defaults:nnnNNn #1#2#3#4#5#6
359  {
360      #6
361      \str_if_eq:nnTF {#1} {#2}
362      { \use_i_delimit_by_q_stop:nw { \group_align_safe_end: #5 #1 } }
363      {
364          \exp_args:No \__cmd_tl_mapthread_function:nnN
365          { #4 #1 } {#3}
366          \__cmd_end_expandable_defaults:nnw
367          \__cmd_end_expandable_defaults:nnnNNn { } {#1} {#3} #4 #5
368      }
369  }
370 \cs_new:Npn \__cmd_end_expandable_defaults:nnw #1#2
371  {
372      \tl_if_novalue:nTF {#2}
373      { \exp_args:No \__cmd_end_expandable_defaults:nw {#1} }
374      { \__cmd_end_expandable_defaults:nw {#2} }
375  }
376 \cs_new:Npn \__cmd_end_expandable_defaults:nw
377      #1#2 \__cmd_end_expandable_defaults:nnnNNn #3
378  { #2 \__cmd_end_expandable_defaults:nnnNNn { #3 {#1} } }

(End of definition for \__cmd_end_expandable:NNw and others.)

```

1.4 Normalizing the argument specifications

The goal here is to expand aliases and check that the argument specification is valid before the main parsing run. If it is not valid the entire set up is abandoned to avoid any strange internal errors. A function is provided for each argument type that will grab any extra data items and call the loop function after performing the following checks and tasks.

- Check that each argument has the correct number of data items associated with it, and that where a single character is required, one has actually been supplied.
- Check that processors and the markers +, ! and = are followed by an argument for which they make sense, and are not redundant.
- Check the absence of forbidden types for expandable commands, namely G/v always, and l/u after optional arguments (`xparse` may have inserted braces due to a failed search for an optional argument).
- Check that no optional argument is followed by a mandatory argument with the same delimiter, as otherwise the optional argument could never be omitted.
- Keep track in `\l__cmd_some_long_bool` and `\l__cmd_some_short_bool` of whether the command has some long/short arguments.

- Keep track in `\l__cmd_grab_expandably_bool` of whether all arguments are `m/l/u` type and short arguments appear before long ones, in which case they can be grabbed expandably just as safely as they could be grabbed nonexpandably. Regardless of that, arguments of expandable commands will be grabbed expandably and arguments of environments will not (because the list of arguments built by non-expandable grabbing is used to pass them to the end-environment code).

Further checks happen at the end of the loop:

- that there are at most 9 arguments;
- that an expandable command does not end with an optional argument (this case is detected by using the fact that `\l__cmd_last_delimiters_tl` is cleared by every mandatory argument and filled by every optional argument).

```
\__cmd_normalize_arg_spec:n Loop through the argument specification, calling an auxiliary specific to each argument
\__cmd_normalize_arg_spec_loop:n type. If any argument is unknown stop the definition.

379 \cs_new_protected:Npn \__cmd_normalize_arg_spec:n #1
380 {
381     \int_zero:N \l__cmd_current_arg_int
382     \tl_clear:N \l__cmd_last_delimiters_tl
383     \tl_clear:N \l__cmd_arg_spec_tl
384     \bool_set_true:N \l__cmd_grab_expandably_bool
385     \bool_set_false:N \l__cmd_obey_spaces_bool
386     \bool_set_false:N \l__cmd_long_bool
387     \bool_set_false:N \l__cmd_suppress_strip_bool
388     \bool_set_false:N \l__cmd_some_obey_spaces_bool
389     \bool_set_false:N \l__cmd_some_long_bool
390     \bool_set_false:N \l__cmd_some_short_bool
391     \__cmd_normalize_arg_spec_loop:n #1
392     \q_recursion_tail \q_recursion_tail \q_recursion_tail \q_recursion_stop
393 \int_compare:nNnT \l__cmd_current_arg_int > 9
394 {
395     \msg_error:nnxx { cmd } { too-many-args }
396     { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
397     \__cmd_bad_def:wn
398 }
399 \bool_if:NT \l__cmd_expandable_bool
400 {
401     \tl_if_empty:NF \l__cmd_last_delimiters_tl
402     {
403         \msg_error:nnxx { cmd } { expandable-ending-optional }
404         { \iow_char:N \\ \l__cmd_function_tl } { \tl_to_str:n {#1} }
405         \__cmd_bad_def:wn
406     }
407 }
408 \bool_if:NT \l__cmd_expandable_bool
409     { \bool_set_true:N \l__cmd_grab_expandably_bool }
410     \bool_if:NT \l__cmd_environment_bool
411     { \bool_set_false:N \l__cmd_grab_expandably_bool }
412 }
413 \cs_new_protected:Npn \__cmd_normalize_arg_spec_loop:n #1
414 {
415     \quark_if_recursion_tail_stop:n {#1}
```

```

416  \int_incr:N \l__cmd_current_arg_int
417  \cs_if_exist_use:cF { __cmd_normalize_type_ \tl_to_str:n {#1} :w }
418  {
419      \bool_lazy_any:nTF
420      {
421          { \str_if_eq_p:nn {#1} { G } }
422          { \str_if_eq_p:nn {#1} { g } }
423          { \str_if_eq_p:nn {#1} { 1 } }
424          { \str_if_eq_p:nn {#1} { u } }
425      }
426      {
427          \msg_error:nnxx { cmd } { xparse-arg-type }
428          { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
429      }
430      {
431          \msg_error:nnxx { cmd } { unknown-argument-type }
432          { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
433      }
434      \__cmd_bad_def:wn
435  }
436 }

```

(End of definition for `__cmd_normalize_arg_spec:n` and `__cmd_normalize_arg_spec_loop:n`.)

`__cmd_normalize_type_d:w`, `__cmd_normalize_type_e:w`, `__cmd_normalize_type_o:w`, `__cmd_normalize_type_0:w`, `__cmd_normalize_type_r:w` and `__cmd_normalize_type_s: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.

```

437 \cs_set_protected:Npn \__cmd_tmp:w #1
438 {
439     \cs_new_protected:Npn \__cmd_normalize_type_d:w ##1##2
440     {
441         \quark_if_recursion_tail_stop_do:nn {##2} { \__cmd_bad_arg_spec:wn }
442         \__cmd_normalize_type_D:w {##1} {##2} {#1}
443     }
444     \cs_new_protected:Npn \__cmd_normalize_type_e:w ##1
445     {
446         \quark_if_recursion_tail_stop_do:nn {##1} { \__cmd_bad_arg_spec:wn }
447         \__cmd_normalize_type_E:w {##1} { }
448     }
449     \cs_new_protected:Npn \__cmd_normalize_type_o:w
450     { \__cmd_normalize_type_D:w [ ] {#1} }
451     \cs_new_protected:Npn \__cmd_normalize_type_0:w
452     { \__cmd_normalize_type_D:w [ ] }
453     \cs_new_protected:Npn \__cmd_normalize_type_r:w ##1##2
454     {
455         \quark_if_recursion_tail_stop_do:nn {##2} { \__cmd_bad_arg_spec:wn }
456         \__cmd_normalize_type_R:w {##1} {##2} {#1}
457     }
458     \cs_new_protected:Npn \__cmd_normalize_type_s:w
459     { \__cmd_normalize_type_t:w * }
460 }
461 \exp_args:No \__cmd_tmp:w { \c_novalue_tl }

```

(End of 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.
`__cmd_normalize_type_+:w`
`__cmd_normalize_type_!/:w`
`__cmd_normalize_type_=:w`
`__cmd_normalize_type_aux:NnNn`

```

462 \cs_new_protected:cpn { __cmd_normalize_type_>:w } #1#2
463 {
464     \quark_if_recursion_tail_stop_do:nn {#2} { __cmd_bad_arg_spec:wn }
465     \bool_if:NT \l__cmd_expandable_bool
466     {
467         \msg_error:nnxx { cmd } { processor-in-expandable }
468         { \iow_char:N \\ \l__cmd_function_tl } { \tl_to_str:n {#1} }
469         \__cmd_bad_def:wn
470     }
471     \tl_put_right:Nx \l__cmd_arg_spec_tl { > { \tl_trim_spaces:n {#1} } }
472     \int_decr:N \l__cmd_current_arg_int
473     \bool_set_false:N \l__cmd_grab_expandably_bool
474     \__cmd_normalize_arg_spec_loop:n {#2}
475 }
476 \cs_new_protected:cpn { __cmd_normalize_type_+:w } #1
477 {
478     \__cmd_normalize_type_aux:NnNn + {#1}
479     \l__cmd_long_bool
480     { \bool_set_true:N \l__cmd_long_bool }
481 }
482 \cs_new_protected:cpn { __cmd_normalize_type_!/:w } #1
483 {
484     \__cmd_normalize_type_aux:NnNn ! {#1}
485     \l__cmd_obey_spaces_bool
486     {
487         \bool_set_true:N \l__cmd_obey_spaces_bool
488         \bool_set_true:N \l__cmd_some_obey_spaces_bool
489     }
490 }
491 \cs_new_protected:cpn { __cmd_normalize_type_=:w } #1#2
492 {
493     \__cmd_normalize_type_aux:NnNn = {#2}
494     \l__cmd_suppress_strip_bool
495     {
496         \bool_if:NT \l__cmd_expandable_bool
497         {
498             \msg_error:nnxx { cmd } { keyval-in-expandable }
499             { \iow_char:N \\ \l__cmd_function_tl } { \tl_to_str:n {#1} }
500             \__cmd_bad_def:wn
501         }
502         \bool_set_true:N \l__cmd_suppress_strip_bool
503         \bool_set_false:N \l__cmd_grab_expandably_bool
504         \tl_put_right:Nx \l__cmd_arg_spec_tl
505         { = { \tl_trim_spaces:n {#1} } }
506     }
507 }
508 \cs_new_protected:Npn \__cmd_normalize_type_aux:NnNn #1#2#3#4
509 {

```

```

510   \quark_if_recursion_tail_stop_do:nn {#2} { \__cmd_bad_arg_spec:wn }
511   \bool_if:NT #3
512   {
513     \msg_error:nnnx { cmd } { two-markers }
514     { \__cmd_environment_or_command: } { #1 }
515     \__cmd_bad_def:wn
516   }
517   #4
518   \int_decr:N \l__cmd_current_arg_int
519   \__cmd_normalize_arg_spec_loop:n {#2}
520 }

```

(End of definition for `__cmd_normalize_type_>:w` and others.)

`__cmd_normalize_type_D:w`
`__cmd_normalize_type_E:w`
`__cmd_normalize_type_t:w`
`__cmd_normalize_E_unique_check: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).

```

521 \cs_new_protected:Npn \__cmd_normalize_type_D:w #1#2#3
522 {
523   \quark_if_recursion_tail_stop_do:nn {#3} { \__cmd_bad_arg_spec:wn }
524   \__cmd_single_token_check:n {#1} \__cmd_allowed_token_check:N #1
525   \__cmd_single_token_check:n {#2}
526   \__cmd_add_arg_spec:n { D #1 #2 {#3} }
527   \tl_put_right:Nn \l__cmd_last_delimiters_tl {#1}
528   \bool_set_false:N \l__cmd_grab_expandably_bool
529   \__cmd_normalize_arg_spec_loop:n
530 }
531 \cs_new_protected:Npn \__cmd_normalize_type_E:w #1#2
532 {
533   \quark_if_recursion_tail_stop_do:nn {#2} { \__cmd_bad_arg_spec:wn }
534   \tl_if_blank:nT {#1} { \__cmd_bad_arg_spec:wn }
535   \tl_map_function:nN {#1} \__cmd_single_token_check:n
536   \tl_map_function:nN {#1} \__cmd_allowed_token_check:N
537   \__cmd_normalize_E_unique_check:w #1 \q_nil \q_stop
538   \int_compare:nNnT { \tl_count:n {#2} } > { \tl_count:n {#1} }
539   { \__cmd_bad_arg_spec:wn }
540   \__cmd_add_arg_spec:n { E {#1} {#2} }
541   \tl_put_right:Nn \l__cmd_last_delimiters_tl {#1}
542   \bool_set_false:N \l__cmd_grab_expandably_bool
543   \int_add:Nn \l__cmd_current_arg_int { \tl_count:n {#1} - 1 }
544   \__cmd_normalize_arg_spec_loop:n
545 }
546 \cs_new_protected:Npn \__cmd_normalize_E_unique_check:w #1#2 \q_stop
547 {
548   \quark_if_nil:NF #1
549   {
550     \tl_if_in:nnT {#2} {#1} { \__cmd_bad_arg_spec:wn }
551     \__cmd_normalize_E_unique_check:w #2 \q_stop
552   }

```

```

553     }
554 \cs_new_protected:Npn \__cmd_normalize_type_t:w #1
555 {
556     \quark_if_recursion_tail_stop_do:Nn #1 { \__cmd_bad_arg_spec:wn }
557     \__cmd_single_token_check:n {#1} \__cmd_allowed_token_check:N #1
558     \tl_put_right:Nx \l__cmd_arg_spec_tl
559     {
560         \bool_if:NT \l__cmd_obey_spaces_bool { ! }
561         t \exp_not:n {#1}
562     }
563     \tl_put_right:Nn \l__cmd_last_delimiters_tl {#1}
564     \bool_set_false:N \l__cmd_grab_expandably_bool
565     \bool_set_false:N \l__cmd_obey_spaces_bool
566     \bool_set_false:N \l__cmd_long_bool
567     \__cmd_normalize_arg_spec_loop:n
568 }

```

(End of definition for `__cmd_normalize_type_D:w` and others.)

`__cmd_normalize_type_m:w`
`__cmd_normalize_type_R:w`
`__cmd_normalize_type_v: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.

```

569 \cs_new_protected:Npn \__cmd_normalize_type_m:w
570 {
571     \__cmd_delimiter_check:nnn { } { m } { \iow_char:N \{ }
572     \__cmd_add_arg_spec_mandatory:n { m }
573     \__cmd_normalize_arg_spec_loop:n
574 }
575 \cs_new_protected:Npn \__cmd_normalize_type_R:w #1#2#3
576 {
577     \quark_if_recursion_tail_stop_do:nn {#3} { \__cmd_bad_arg_spec:wn }
578     \__cmd_single_token_check:n {#1} \__cmd_allowed_token_check:N #1
579     \__cmd_single_token_check:n {#2}
580     \__cmd_delimiter_check:nnn {#1} { R/r } { \tl_to_str:n {#1} }
581     \bool_set_false:N \l__cmd_grab_expandably_bool
582     \__cmd_add_arg_spec_mandatory:n { R #1 #2 {#3} }
583     \__cmd_normalize_arg_spec_loop:n
584 }
585 \cs_new_protected:Npn \__cmd_normalize_type_v:w
586 {
587     \__cmd_normalize_check_gv:N v
588     \__cmd_add_arg_spec_mandatory:n { v }
589     \__cmd_normalize_arg_spec_loop:n
590 }

```

(End of 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.

```
591 \cs_new_protected:Npn \__cmd_normalize_type_b:w #1
```

```

592  {
593    \bool_if:NF \l__cmd_environment_bool
594    {
595      \msg_error:nnxx { cmd } { invalid-command-arg }
596      { \__cmd_environment_or_command: } { b }
597      \__cmd_bad_def:wn
598    }
599    \tl_clear:N \l__cmd_last_delimiters_tl
600    \__cmd_add_arg_spec:n { b }
601    \quark_if_recursion_tail_stop:n {#1}
602    \msg_error:nnxx { cmd } { arg-after-body }
603    { \__cmd_environment_or_command: }
604    { \tl_to_str:n {#1} }
605    \__cmd_bad_def:wn
606  }

```

(End of 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.

```

607 \cs_new_protected:Npn \__cmd_single_token_check:n #1
608  {
609    \tl_trim_spaces_apply:nN {#1} \tl_if_single_token:nF
610    {
611      \msg_error:nnxx { cmd } { not-single-token }
612      { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
613      \__cmd_bad_def:wn
614    }
615  }

```

(End of definition for `__cmd_single_token_check:n.`)

`__cmd_allowed_token_check:N` Some tokens are not 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 `}2`, leading to an ! Argument of `\dots` has an extra `}` error.

```

616 \cs_new_protected:Npn \__cmd_allowed_token_check:N #1
617  {
618    \token_if_eq_meaning:NNTF #1 \c_group_begin_token
619    { \use:n }
620    {
621      \token_if_eq_meaning:NNTF #1 \c_group_end_token
622      { \use:n }
623      { \use_none:n }
624    }
625  {
626    \msg_error:nnxxx { cmd } { forbidden-group-token }
627    { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
628    {

```

```

629         \token_if_eq_meaning:NNTF #1 \c_group_begin_token
630             { begin } { end }
631         }
632     \__cmd_bad_def:wn
633 }
634 }
```

(End of definition for __cmd_allowed_token_check:N.)

__cmd_normalize_check_gv:N Called for arguments that are always forbidden, or forbidden after an optional argument,
__cmd_normalize_check_lu:N for expandable commands.

```

635 \cs_new_protected:Npn \__cmd_normalize_check_gv:N #1
636 {
637     \bool_if:NT \l__cmd_expandable_bool
638     {
639         \msg_error:nnxx { cmd } { invalid-expandable-arg }
640         { \iow_char:N \\ \l__cmd_function_tl } { \tl_to_str:n {#1} }
641         \__cmd_bad_def:wn
642     }
643     \bool_set_false:N \l__cmd_grab_expandably_bool
644 }
645 \cs_new_protected:Npn \__cmd_normalize_check_lu:N #1
646 {
647     \bool_if:NT \l__cmd_expandable_bool
648     {
649         \tl_if_empty:NF \l__cmd_last_delimiters_tl
650         {
651             \msg_error:nnxx { cmd } { invalid-after-optional-expandably }
652             { \iow_char:N \\ \l__cmd_function_tl } { \tl_to_str:n {#1} }
653             \__cmd_bad_def:wn
654         }
655     }
656 }
```

(End of 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.

```

657 \cs_new_protected:Npn \__cmd_delimiter_check:nnn #1#2#3
658 {
659     \tl_map_inline:Nn \l__cmd_last_delimiters_tl
660     {
661         \tl_if_eq:nnT {##1} {#1}
662         {
663             \msg_warning:nnxx { cmd } { optional-mandatory }
664             {#2} {#3}
665         }
666     }
667 }
```

(End of definition for __cmd_delimiter_check:nnn.)

__cmd_bad_arg_spec:wn If the argument specification is wrong, this provides an escape from the entire definition process.

```

668 \cs_new_protected:Npn \_\_cmd_bad_arg_spec:wn #1 \_\_cmd_break_point:n #2
669  {
670      \msg_error:nnnx { cmd } { bad-arg-spec }
671      { \_\_cmd_environment_or_command: } { \tl_to_str:n {#2} }
672  }
673 \cs_new_protected:Npn \_\_cmd_bad_def:wn #1 \_\_cmd_break_point:n #2 { }

(End of definition for \_\_cmd_bad_arg_spec:wn and \_\_cmd_bad_def:wn.)
```

__cmd_add_arg_spec: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 !.

```

674 \cs_new_protected:Npn \_\_cmd_add_arg_spec:n #1
675  {
676      \bool_lazy_and:nnT
677      { ! \l__cmd_long_bool }
678      { \l__cmd_some_long_bool }
679      {
680          \bool_if:NT \l__cmd_expandable_bool
681          {
682              \msg_error:nnx { cmd } { long-short-mix }
683              { \iow_char:N \\ \l__cmd_function_tl }
684              \_\_cmd_bad_def:wn
685          }
686          \bool_set_false:N \l__cmd_grab_expandably_bool
687      }
688      \bool_if:NTF \l__cmd_long_bool
689      { \bool_set_true:N \l__cmd_some_long_bool }
690      { \bool_set_true:N \l__cmd_some_short_bool }
691      \tl_put_right:Nx \l__cmd_arg_spec_tl
692      {
693          \bool_if:NT \l__cmd_long_bool { + }
694          \bool_if:NT \l__cmd_obey_spaces_bool { ! }
695          \exp_not:n {#1}
696      }
697      \bool_set_false:N \l__cmd_long_bool
698      \bool_set_false:N \l__cmd_obey_spaces_bool
699  }
700 \cs_new_protected:Npn \_\_cmd_add_arg_spec_mandatory:n #1
701  {
702      \bool_if:NT \l__cmd_some_obey_spaces_bool
703      {
704          \msg_error:nnnx { cmd } { invalid-bang }
705          { \_\_cmd_environment_or_command: } { \tl_to_str:n {#1} }
706          \_\_cmd_bad_def:wn
707      }
708      \tl_clear:N \l__cmd_last_delimiters_tl
709      \_\_cmd_add_arg_spec:n {#1}
710 }
```

(End of 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.

```
711 \cs_new_protected:Npn \_cmd_prepare_signature:n #1
712 {
713     \int_zero:N \l__cmd_current_arg_int
714     \bool_set_false:N \l__cmd_long_bool
715     \bool_set_false:N \l__cmd_obey_spaces_bool
716     \bool_set_false:N \l__cmd_suppress_strip_bool
717     \int_zero:N \l__cmd_m_args_int
718     \bool_set_false:N \l__cmd_defaults_bool
719     \tl_clear:N \l__cmd_defaults_tl
720     \tl_clear:N \l__cmd_process_all_tl
721     \tl_clear:N \l__cmd_process_one_tl
722     \bool_set_false:N \l__cmd_process_some_bool
723     \tl_clear:N \l__cmd_signature_tl
724     \_cmd_prepare_signature:N #1 \q_recursion_tail \q_recursion_stop
725     \bool_if:NF \l__cmd_expandable_bool { \_cmd_flush_m_args: }
726 }
```

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.

```
727 \cs_new_protected:Npn \_cmd_prepare_signature:N
728 {
729     \bool_set_false:N \l__cmd_prefixed_bool
730     \_cmd_prepare_signature_bypass:N
731 }
732 \cs_new_protected:Npn \_cmd_prepare_signature_bypass:N #1
733 {
734     \quark_if_recursion_tail_stop:N #1
735     \use:c
736     {
737         \__cmd_add
738         \bool_if:NT \l__cmd_grab_expandably_bool { _expandable }
739         _type_ \token_to_str:N #1 :w
740     }
741 }
```

(End of 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.

```

742 \cs_new_protected:cpn { __cmd_add_type_+::w }
743 {
744     __cmd_flush_m_args:
745     \bool_set_true:N \l__cmd_long_bool
746     \bool_set_true:N \l__cmd_prefixed_bool
747     \__cmd_prepare_signature_bypass:N
748 }
```

(End of definition for `__cmd_add_type_+::w`.)

`__cmd_add_type_!::w` Much the same for controlling trailing optional arguments.

```

749 \cs_new_protected:cpn { __cmd_add_type_!::w }
750 {
751     __cmd_flush_m_args:
752     \bool_set_true:N \l__cmd_obey_spaces_bool
753     \bool_set_true:N \l__cmd_prefixed_bool
754     \__cmd_prepare_signature_bypass:N
755 }
```

(End of 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.

```

756 \cs_new_protected:cpn { __cmd_add_type_>::w } #1
757 {
758     __cmd_flush_m_args:
759     \bool_set_true:N \l__cmd_prefixed_bool
760     \bool_set_true:N \l__cmd_process_some_bool
761     \tl_put_left:Nn \l__cmd_process_one_tl { {#1} }
762     \__cmd_prepare_signature_bypass:N
763 }
```

(End of definition for `__cmd_add_type_>::w`.)

`__cmd_add_type_=:` A mix of the ideas from above: set a flag and add a processor.

```

764 \cs_new_protected:cpn { __cmd_add_type_=::w } #1
765 {
766     __cmd_flush_m_args:
767     \bool_set_true:N \l__cmd_prefixed_bool
768     \bool_set_true:N \l__cmd_suppress_strip_bool
769     \bool_set_true:N \l__cmd_process_some_bool
770     \tl_put_left:Nn \l__cmd_process_one_tl
771     { { \__cmd_arg_to_keyvalue:nn {#1} } }
772     \__cmd_prepare_signature_bypass:N
773 }
```

(End of definition for `__cmd_add_type_=:.`)

```
\_\_cmd\_add\_type\_b:w
774 \cs_new_protected:Npn \_\_cmd\_add\_type\_b:w
775 {
776     \_\_cmd\_flush_m\_args:
777     \_\_cmd\_add\_default:
778     \_\_cmd\_add\_grabber:N b
779     \_\_cmd\_prepare\_signature:N
780 }
```

(End of definition for `__cmd_add_type_b:w.`)

```
\_\_cmd\_add\_type\_D:w
781 \cs_new_protected:Npn \_\_cmd\_add\_type\_D:w #1#2#3
782 {
783     \_\_cmd\_flush_m\_args:
784     \_\_cmd\_add\_default:n {#3}
785     \_\_cmd\_add\_grabber:N D
786     \tl_put_right:Nn \l_\_cmd\_signature_tl { #1 #2 }
787     \_\_cmd\_prepare\_signature:N
788 }
```

(End of 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.

```
789 \cs_new_protected:Npn \_\_cmd\_add\_type\_E:w #1#2
790 {
791     \_\_cmd\_flush_m\_args:
792     \_\_cmd\_add\_default_E:nn {#1} {#2}
793     \use:x
794     {
795         \_\_cmd\_replicate_processor:nn { \tl_count:n {#1} }
796         { \exp_not:o \l_\_cmd\_process\_one_tl }
797     }
798     \_\_cmd\_add\_grabber:N E
799     \tl_put_right:Nn \l_\_cmd\_signature_tl { {#1} }
800     \_\_cmd\_prepare\_signature:N
801 }
```

(End of 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.

```

802 \cs_new_protected:Npn \__cmd_replicate_processor:nn #1 #2
803 {
804     \int_compare:nNnF {#1} > { 1 } { \use_none:nnn }
805     \tl_set:Nx \l__cmd_process_one_tl
806     {
807         \exp_not:n { \exp_not:n {#2} \if_false: { \fi: } }
808         \prg_replicate:nn {#1 - 2}
809         { \exp_not:n { \exp_not:n { {#2} } } }
810         \exp_not:n { { \if_false: } \fi: \exp_not:n {#2} }
811     }
812 }

```

(End of 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:`.

```

813 \cs_new_protected:Npn \__cmd_add_type_m:w
814 {
815     \__cmd_add_default:
816     \bool_if:NTF \l__cmd_prefixed_bool
817     { \__cmd_add_grabber:N m }
818     { \int_incr:N \l__cmd_m_args_int }
819     \__cmd_prepare_signature:N
820 }

```

(End of definition for `__cmd_add_type_m:w`.)

`__cmd_add_type_R:w` The R-type argument is very similar to the D-type.

```

821 \cs_new_protected:Npn \__cmd_add_type_R:w #1#2#3
822 {
823     \__cmd_flush_m_args:
824     \__cmd_add_default:n {#3}
825     \__cmd_add_grabber:N R
826     \tl_put_right:Nn \l__cmd_signature_tl { #1 #2 }
827     \__cmd_prepare_signature:N
828 }

```

(End of 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.

```
829 \cs_new_protected:Npn \_\_cmd\_add\_type\_t:w #1
830   {
831     \_\_cmd\_flush_m\_args:
832     \_\_cmd\_add\_default:
833     \_\_cmd\_add\_grabber:N t
834     \tl_put_right:Nn \l_\_cmd\_signature_tl {\#1}
835     \_\_cmd\_prepare\_signature:N
836   }
```

(End of 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.

```
837 \cs_new_protected:Npn \_\_cmd\_add\_type\_v:w
838   {
839     \_\_cmd\_flush_m\_args:
840     \exp_args:No \_\_cmd\_add\_default:n \c_novalue_tl
841     \_\_cmd\_add\_grabber:N v
842     \_\_cmd\_prepare\_signature:N
843   }
```

(End of 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.

```
844 \cs_new_protected:Npn \_\_cmd\_flush_m\_args:
845   {
846     \int_compare:nNnT \l_\_cmd_m\_args_int > 0
847     {
848       \tl_put_right:Nx \l_\_cmd\_signature_tl
849       { \exp_not:c { \_\_cmd\_grab_m\_ \int_use:N \l_\_cmd_m\_args_int :w } }
850       \tl_put_right:Nx \l_\_cmd\_process\_all_tl
851       { \prg_replicate:nn { \l_\_cmd_m\_args_int } { { } } }
852     }
853     \int_zero:N \l_\_cmd_m\_args_int
854   }
```

(End of 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.

```
855 \cs_new_protected:Npn \_\_cmd\_add\_grabber:N #1
856   {
857     \tl_put_right:Nx \l_\_cmd\_signature_tl
858     {
859       \exp_not:c
```

```

860         {
861             __cmd_grab_ #1
862             \bool_if:NT \l__cmd_long_bool { _long }
863             \bool_if:NT \l__cmd_obey_spaces_bool { _obey_spaces }
864             \bool_lazy_and:nnT
865                 { \l__cmd_suppress_strip_bool }
866                 { \str_if_eq_p:nn {#1} { D } }
867                 { _no_strip }
868             :w
869         }
870     }
871     \bool_set_false:N \l__cmd_long_bool
872     \bool_set_false:N \l__cmd_obey_spaces_bool
873     \bool_set_false:N \l__cmd_suppress_strip_bool
874     \tl_put_right:Nx \l__cmd_process_all_tl
875     {
876         {
877             \if_charcode:w E #1 \use_i:nn \fi:
878             \exp_not:o \l__cmd_process_one_tl
879         }
880     }
881     \tl_clear:N \l__cmd_process_one_tl
882 }

```

(End of 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.

```

883 \cs_new_protected:Npn \__cmd_add_default:n #1
884 {
885     \tl_if_novalue:nTF {#1}
886     { \__cmd_add_default: }
887     {
888         \int_incr:N \l__cmd_current_arg_int
889         \bool_set_true:N \l__cmd_defaults_bool
890         \tl_put_right:Nn \l__cmd_defaults_tl { { \prg_do_nothing: #1 } }
891     }
892 }
893 \cs_new_protected:Npn \__cmd_add_default:
894 {
895     \int_incr:N \l__cmd_current_arg_int
896     \tl_put_right:Nn \l__cmd_defaults_tl { \c_novalue_tl }
897 }
898 \cs_new_protected:Npn \__cmd_add_default_E:nn #1#2
899 {
900     \tl_map_function:nN {#2} \__cmd_add_default:n
901     \prg_replicate:nn
902         { \tl_count:n {#1} - \tl_count:n {#2} }
903         { \__cmd_add_default: }
904 }

```

(End of 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.

```
905 \cs_new_protected:cpn { \_\_cmd\_add\_expandable\_type_+:w }
906   {
907     \bool_set_true:N \l\_\_cmd\_long\_bool
908     \_\_cmd_prepare_signature:N
909   }
```

(End of 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.

```
910 \cs_new_protected:Npn \_\_cmd\_add\_expandable\_type_D:w
911   { \_\_cmd\_add\_expandable\_type_D\_aux:NNNn D }
912 \cs_new_protected:Npn \_\_cmd\_add\_expandable\_type_D\_aux:NNNn #1#2#3#4
913   {
914     \_\_cmd\_add\_default:n {#4}
915     \tl_if_eq:nnTF {#2} {#3}
916       { \_\_cmd\_add\_expandable\_type_D\_aux:NN #1 #2 }
917       { \_\_cmd\_add\_expandable\_type_D\_aux:NNN #1 #2 #3 }
918     \_\_cmd_prepare_signature:N
919   }
920 \cs_new_protected:Npn \_\_cmd\_add\_expandable\_type_D\_aux:NNN #1#2#3
921   {
922     \bool_if:NTF \l\_\_cmd\_long\_bool
923       { \cs_set:cpx }
924       { \cs_set_nopar:cpx }
925       { \l\_\_cmd_expandable_aux_name_tl } ##1 ##2 #2 ##3 \q\_\_cmd ##4 #3
926       { ##1 {##2} {##3} {##4} }
927     \_\_cmd\_add\_expandable_grabber:nn {#1}
928     {
929       \exp_not:c { \l\_\_cmd_expandable_aux_name_tl }
930       \exp_not:n { #2 #3 }
931     }
932   }
933 \cs_new_protected:Npn \_\_cmd\_add\_expandable\_type_D\_aux:NN #1#2
934   {
935     \bool_if:NTF \l\_\_cmd\_long\_bool
936       { \cs_set:cpx }
937       { \cs_set_nopar:cpx }
938       { \l\_\_cmd_expandable_aux_name_tl } ##1 #2 ##2 #2
```

```

939   { ##1 {##2} }
940   \__cmd_add_expandable_grabber:nn { #1_alt }
941   {
942     \exp_not:c { \l__cmd_expandable_aux_name_t1 }
943     \exp_not:n {#2}
944   }
945 }
```

(End of 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.

```

946 \cs_new_protected:Npn \__cmd_add_expandable_type_E:w #1#2
947   {
948     \__cmd_add_default_E:nn {#1} {#2}
949     \tl_clear:N \l__cmd_tmpb_t1
950     \tl_map_function:nN {#1} \__cmd_add_expandable_type_E_aux:n
951     \__cmd_add_expandable_grabber:nn
952     { E \bool_if:NT \l__cmd_long_bool { _long } }
953     {
954       { \exp_not:o \l__cmd_tmpb_t1 }
955       {
956         \prg_replicate:nn { \tl_count:n {#1} }
957         { { \c_novalue_t1 } }
958       }
959     }
960     \__cmd_prepare_signature:N
961   }
962 \cs_new_protected:Npn \__cmd_add_expandable_type_E_aux:n #1
963   {
964     \__cmd_get_grabber:NN #1 \l__cmd_tmpa_t1
965     \tl_put_right:Nx \l__cmd_tmpb_t1
966     { \exp_not:o \l__cmd_tmpa_t1 \exp_not:N #1 }
967 }
```

(End of 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.

```

968 \cs_new_protected:Npn \__cmd_add_expandable_type_m:w
969   {
970     \__cmd_add_default:
971     \__cmd_add_expandable_grabber:nn
972     { m \bool_if:NT \l__cmd_long_bool { _long } } { }
973     \__cmd_prepare_signature:N
974   }
```

(End of 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.

```

975 \cs_new_protected:Npn \__cmd_add_expandable_type_R:w
976   { \__cmd_add_expandable_type_D_aux:NNNn R }
```

(End of 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.

```
977 \cs_new_protected:Npn \_cmd_add_expandable_type_t:w #1
978 {
979     \_cmd_add_default:
980     \_cmd_get_grabber>NN #1 \l__cmd_tmpa_tl
981     \_cmd_add_expandable_grabber:nn { t }
982     {
983         \exp_not:o \l__cmd_tmpa_tl
984         \exp_not:N #1
985     }
986     \_cmd_prepare_signature:N
987 }
```

(End of 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.

```
988 \cs_new_protected:Npn \_cmd_add_expandable_grabber:nn #1#2
989 {
990     \tl_put_right:Nx \l__cmd_signature_tl
991     { \exp_not:c { __cmd_expandable_grab_ #1 :w } #2 }
992 }
```

(End of definition for `_cmd_add_expandable_grabber:nn`.)

`_cmd_get_grabber>NN
_cmd_get_grabber_auxi>NN
_cmd_get_grabber_auxii>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.

```
993 \cs_new_protected:Npn \_cmd_get_grabber>NN #1#2
994 {
995     \cs_set:Npn \_cmd_tmp:w ##1 #1 {##1}
996     \exp_args:Nc \_cmd_get_grabber_auxi>NN
997     { __cmd_grabber_ \token_to_str:N #1 :w } #2
998 }
999 \cs_new_protected:Npn \_cmd_get_grabber_auxi>NN #1#2
1000 {
1001     \cs_if_eq:NNTF \_cmd_tmp:w #1
1002     { \tl_set:Nn #2 {#1} }
1003     {
1004         \cs_if_exist:NTF #1
1005         {
1006             \int_gincr:N \g__cmd_grabber_int
1007             \exp_args:Nc \_cmd_get_grabber_auxi>NN
1008             {
1009                 __cmd_grabber_
1010                 - \int_use:N \g__cmd_grabber_int :w
1011             }
1012             #2
1013         }
1014 }
```

```

1014         { \__cmd_get_grabber_auxii:NN #1 #2 }
1015     }
1016 }
1017 \cs_new_protected:Npn \__cmd_get_grabber_auxii:NN #1#2
1018 {
1019     \cs_set_eq:NN #1 \__cmd_tmp:w
1020     \tl_set:Nn #2 {#1}
1021 }

```

(End of definition for `__cmd_get_grabber:NN`, `__cmd_get_grabber_auxi:NN`, and `__cmd_get_grabber_auxii:NN`.)

1.7.1 Copying a command and its internal structure

```

1022 <latexrelease> \IncludeInRelease{2021/11/15}{\__cmd_copy:NN}%
1023 <latexrelease> {Support-\NewCommandCopy-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.

```

1024 \cs_new_protected:Npn \__cmd_copy:NN #1 #2
1025 {
1026     \use:x
1027     {
1028         \int_set:Nn \tex_escapechar:D { 92 }
1029         \exp_not:N \__cmd_cmd_type_cases:NnnnnF \exp_not:N #2
1030         { \__cmd_copy_command:nnNN }
1031         { \__cmd_copy_expandable:nnNN }
1032         { \__cmd_copy_environment:nnNN }
1033         { \__cmd_copy_environment_end:nnNN }
1034         { \__cmd_cant_copy:nwn { non-ltcmd } }
1035         { \cs_to_str:N #1 } { \cs_to_str:N #2 }
1036         \exp_not:N #1 \exp_not:N #2
1037         \exp_not:N \__cmd_break_point:n { \cs_to_str:N #2 }
1038         \int_set:Nn \tex_escapechar:D { \int_use:N \tex_escapechar:D }
1039     }
1040 }
1041 \cs_new_protected:Npn \__cmd_set_eq_if_exist:NN #1 #2
1042 {
1043     \cs_if_exist:NTF #2 { \cs_set_eq:NN } { \use_none:nn } #1 #2
1044 \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.

```

1044 \cs_new_protected:Npn \__cmd_cant_copy:nwn #1 #2 \__cmd_break_point:n #3
1045   { \msg_error:nnnn { cmd } { copy-bug } {#1} {#3} }
1046 \msg_new:nnn { cmd } { copy-bug }
1047   {
1048     Error~while~copying~command~\iow_char:N\\#2:\\
1049     \str_case:nn {#1}
1050       {
1051         { non-ltcmd } { Command~is~not~a~valid~ltcmd~command. }
1052         { unknown-type } { Found~an~unknown~argument~type. }
1053         { invalid-end }
1054         { Target~command~is~not~named~\iow_char:N \\end<name>. }
1055       }
1056     }

```

And, of course, add `__kernel_cmd_if_xparse:NTF` and `__cmd_copy:NN` to `\@declarecommandcopylisthook`:

```

1057 \tl_gput_right:Nn \@declarecommandcopylisthook
1058   { { \__kernel_cmd_if_xparse:NTF \__cmd_copy:NN } }

```

(End of definition for `__cmd_copy:NN`, `__cmd_set_eq_if_exist:NN`, and `__cmd_cant_copy:nwn`.)

`__cmd_copy_command:nnNN` and `__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.

```

1059 \cs_new_protected:Npn \__cmd_copy_command:nnNN #1 #2 #3 #4
1060   {
1061     \cs_set_eq:cc { #1 ~ code } { #2 ~ code }
1062     \__cmd_set_eq_if_exist:cc { #1 ~ defaults } { #2 ~ defaults }
1063     \cs_set_protected_nopar:Npx #3
1064     { \exp_after:wN \__cmd_copy_command:NnNNnnnn #4 {#1} }
1065   }
1066 \cs_new:Npn \__cmd_copy_command:NnNNnnnn #1 #2 #3 #4 #5 #6 #7 #8
1067   {
1068     #1 \exp_not:n { {#2} }
1069     \exp_not:c { #8 ~ } \exp_not:c { #8 ~ code }
1070     \exp_not:n { {#5} {#6} {#7} }
1071   }

```

(End of definition for `__cmd_copy_command:nnNN` and `__cmd_copy_command:NnNNnnnn`.)

`__cmd_copy_expandable:nnNN` and `__cmd_copy_expandable:NnNNNNnn`

An expandable command is slightly more complicated. Besides the `\⟨cmd⟩_code`, and `\⟨cmd⟩_defaults`, it also has an auxiliary `\⟨cmd⟩_L` for grabbing delimited arguments, and possibly another auxiliary `\⟨cmd⟩_L`, 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:NnNNNNnn` to parse the signature of the command and build up the modified copy in a temporary token list, then we call `__cmd_copy_expandable:NnNNNNnn` that will copy the top-level definition of the command, with the proper internal renames.

```

1072 <|latexrelease>\EndIncludeInRelease
1073 <|latexrelease>\IncludeInRelease{2020/10/01}{\_\_cmd\_copy:NN}%
1074 <|latexrelease> {Support~\NewCommandCopy~in~ltcmd}
1075 <|latexrelease>\EndIncludeInRelease

    There's one variant: a command begins with \_\_cmd\_start\_expandable:nNNNNn
may still be un-expandable/protected if it's defined by \NewDocumentCommand and
friends, with empty or only m-type arguments.

1076 <|latexrelease>\IncludeInRelease{2023/06/01}{\_\_cmd\_copy\_expandable:nnNN}%
1077 <|latexrelease> {Distinguish~non~expandable~document~commands}
1078 \cs_new_protected:Npn \_\_cmd_copy_expandable:nnNN #1 #2 #3 #4
1079 {
1080     \cs_set_eq:cc { #1 ~ code } { #2 ~ code }
1081     \_\_cmd_set_eq_if_exist:cc { #1 ~ } { #2 ~ }
1082     \_\_cmd_set_eq_if_exist:cc { #1 ~ \c_space_tl } { #2 ~ \c_space_tl }
1083     \_\_cmd_set_eq_if_exist:cc { #1 ~ defaults } { #2 ~ defaults }
1084     \exp_after:wN \_\_cmd_copy_expandable_signature:NnNNNNnnn #4 {#1} {#2}
1085     \token_if_protected_macro:NTF #4
1086         { \cs_set_protected_nopar:Npx }{ \cs_set_nopar:Npx }
1087         #3
1088         { \exp_after:wN \_\_cmd_copy_expandable:NnNNNNnnn #4 {#1} {#2} }
1089     }
1090 <|latexrelease>\EndIncludeInRelease
1091 <|latexrelease>\IncludeInRelease{2021/11/15}{\_\_cmd\_copy\_expandable:nnNN}%
1092 <|latexrelease> {Support~\NewCommandCopy~in~ltcmd}
1093 \cs_new_protected:Npn \_\_cmd_copy_expandable:nnNN #1 #2 #3 #4
1094 <|latexrelease> {
1095     \cs_set_eq:cc { #1 ~ code } { #2 ~ code }
1096     \_\_cmd_set_eq_if_exist:cc { #1 ~ } { #2 ~ }
1097     \_\_cmd_set_eq_if_exist:cc { #1 ~ \c_space_tl } { #2 ~ \c_space_tl }
1098     \_\_cmd_set_eq_if_exist:cc { #1 ~ defaults } { #2 ~ defaults }
1099     \exp_after:wN \_\_cmd_copy_expandable_signature:NnNNNNnnn #4 {#1} {#2}
1100     \cs_set_nopar:Npx #3
1101     { \exp_after:wN \_\_cmd_copy_expandable:NnNNNNnnn #4 {#1} {#2} }
1102 <|latexrelease> }
1103 <|latexrelease>\EndIncludeInRelease
1104 <|latexrelease>\IncludeInRelease{2020/10/01}{\_\_cmd\_copy\_expandable:nnNN}%
1105 <|latexrelease> {Support~\NewCommandCopy~in~ltcmd}
1106 <|latexrelease>\EndIncludeInRelease

1107 <|latexrelease>\IncludeInRelease{2021/11/15}{\_\_cmd\_copy:NN (part 2)}%
1108 <|latexrelease> {Support~\NewCommandCopy~in~ltcmd}

1109 \cs_new:Npn \_\_cmd_copy_expandable:NnNNNNnnn #1 #2 #3 #4 #5 #6 #7 #8 #9
1110 {
1111     \exp_not:N #1 \exp_not:n { {#2} }
1112     \exp_not:c { #8 ~ }
1113     \exp_not:c
1114     {
1115         #8 ~
1116         \str_if_eq:eeT
1117             { \exp_not:c { #9 ~ \c_space_tl } } { \exp_not:N #4 }
1118             { \c_space_tl }
1119     }
1120     \exp_not:c { #8 ~ code }

```

```

1121     \str_if_eq:eeTF { \exp_not:N #6 } { ? }
1122     { ? }
1123     { \exp_not:c { #8 ~ defaults } }
1124     { \exp_not:V \l__cmd_tmpa_t1 }
1125 }
```

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.

```

1126 \cs_new_protected:Npn \__cmd_copy_expandable_signature:NnNNNNnnn
1127   #1 #2 #3 #4 #5 #6 #7 #8 #9
1128 {
1129   \int_zero:N \l__cmd_current_arg_int
1130   \tl_clear:N \l__cmd_tmpa_t1
1131   \__cmd_copy_expandable:nnN {#8} {#9} #7
1132   \q_recursion_tail \q_recursion_stop
1133 }
1134 \cs_new_protected:Npn \__cmd_copy_expandable:nnN #1 #2 #3
1135 {
1136   \quark_if_recursion_tail_stop:n {#3}
1137   \int_incr:N \l__cmd_current_arg_int
1138   \exp_after:wN \__cmd_copy_parse_grabber:w \token_to_str:N #3 {#1} {#2}
1139 }
1140 \use:x
1141 {
1142   \cs_new_protected:Npn \exp_not:N \__cmd_copy_parse_grabber:w ##1
1143     \tl_to_str:n { expandable_grab_ } ##2 \tl_to_str:n { :w }
1144   {
1145     \tl_put_right:Nx \exp_not:N \l__cmd_tmpa_t1
1146     { \exp_not:N \exp_not:c { __cmd_expandable_grab_##2:w } }
1147     \exp_not:N \cs_if_exist_use:cF { __cmd_copy_grabber_##2:w }
1148     { \__cmd_cant_copy:nwn { unknown-type } }
1149   }
1150 }
```

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 `\(cmd)_<arg>\(num)`).

```

1151 \cs_new_protected:Npn \__cmd_copy_grabber_D:w #1 #2 #3 #4 #5
1152 {
1153   \tl_put_right:Nx \l__cmd_tmpa_t1
1154   {
1155     \exp_not:c { #1 ~ (arg ~ \int_use:N \l__cmd_current_arg_int ) }
1156     \exp_not:n { #4 #5 }
1157   }
1158   \cs_set_eq:cc
1159   { #1 ~ (arg ~ \int_use:N \l__cmd_current_arg_int ) }
1160   { #2 ~ (arg ~ \int_use:N \l__cmd_current_arg_int ) }
1161   \__cmd_copy_expandable:nnN {#1} {#2}
1162 }
```

`D_alt` is just a special case of `D` that uses a single delimiter (used when both delimiters of the argument are identical):

```
1163 \cs_new_protected:Npn \__cmd_copy_grabber_D_alt:w #1 #2 #3 #4
1164   { \__cmd_copy_grabber_D:w {#1} {#2} {#3} {#4} { } }
```

As far as copying is concerned, `R` is identical to `D`:

```
1165 \cs_new_eq:NN \__cmd_copy_grabber_R:w \__cmd_copy_grabber_D:w
1166 \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).

```
1167 \cs_new_protected:Npn \__cmd_copy_grabber_E:w #1 #2 #3 #4
1168   {
1169     \tl_put_right:Nn \l__cmd_tmpa_tl { {#3} {#4} }
1170     \int_add:Nn \l__cmd_current_arg_int { \tl_count:n {#4} - 1 }
1171     \__cmd_copy_expandable:nnN {#1} {#2}
1172   }
1173 \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:

```
1174 \cs_new_protected:Npn \__cmd_copy_grabber_t:w #1 #2 #3 #4
1175   {
1176     \tl_put_right:Nn \l__cmd_tmpa_tl { #3 #4 }
1177     \__cmd_copy_expandable:nnN {#1} {#2}
1178   }
```

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:

```
1179 \cs_new_protected:Npn \__cmd_copy_grabber_m:w { \__cmd_copy_expandable:nnN }
1180 \cs_new_eq:NN \__cmd_copy_grabber_m_long:w \__cmd_copy_grabber_m:w
```

(End of definition for `__cmd_copy_expandable:nnNN` and others.)

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 <name>` into `\<name>`.

```
1181 \cs_new_protected:Npn \__cmd_copy_environment:nnNN #1 #2 #3 #4
1182   {
1183     \cs_set_eq:cc { environment~ #1 ~ code } { environment~ #2 ~ code }
1184     \__cmd_set_eq_if_exist:cc
1185       { environment~ #1 ~ defaults } { environment~ #2 ~ defaults }
1186     \cs_set_protected_nopar:cpx { environment~ #1 }
1187       { \exp_after:wN \__cmd_copy_environment:Nnnnnnn #4 {#1} }
1188     \cs_set_eq:cc {#1} { environment~ #1 }
1189   }
1190 \cs_new:Npn \__cmd_copy_environment:Nnnnnnn #1 #2 #3 #4 #5 #6 #7
1191   { #1 \exp_not:n { {#2} } {#7} \exp_not:n { {#4} {#5} {#6} } }
```

(End of definition for `__cmd_copy_environment:nnNN` and `__cmd_copy_environment:Nnnnnnn`.)

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.

```
1192 \cs_new_protected:Npn \__cmd_copy_environment_end:nnNN #1 #2
```

```

1193   {
1194     \__cmd_check_end:Nn \l__cmd_tmpa_t1 {#1}
1195     \__cmd_check_end:Nn \l__cmd_tmpb_t1 {#2}
1196     \exp_args:Nno \__cmd_copy_environment_end_aux:nnNN
1197     { \l__cmd_tmpa_t1 } { \l__cmd_tmpb_t1 }
1198   }
1199 \cs_new_protected:Npn \__cmd_copy_environment_end_aux:nnNN #1 #2 #3 #4
1200   {
1201     \cs_set_nopar:cp { environment~ #1 ~end }
1202     { \exp_not:c { environment~ #1 ~end~aux } }
1203     \cs_set_eq:cc
1204     { environment~ #1 ~end~aux~ } { environment~ #2 ~end~aux~ }
1205     \cs_set_eq:cc { end #1 } { environment~ #1 ~end }
1206   }

```

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
1207 \cs_new_protected:Npn \__cmd_check_end:Nn #1 #2
\__cmd_check_end:n
1208   {
1209     \tl_set:Nx #1 { \__cmd_check_end:n {#2} }
1210     \token_if_eq_meaning:NNT #1 \q_nil
1211     { \__cmd_cant_copy:nwn { invalid-end } }
1212   }
1213 \cs_set_protected:Npn \__cmd_tmp:w #1
1214   {
1215     \cs_new:Npn \__cmd_check_end:n ##1
1216     {
1217       \exp_after:wN \__cmd_check_end:w \tl_to_str:n {##1}
1218       #1 \q_mark #1 \q_stop
1219     }
1220     \cs_new:Npn \__cmd_check_end:w ##1 #1 ##2 #1 ##3 \q_stop
1221     { \if_meaning:w ##2 \q_mark \exp_not:N \q_nil \else: ##2 \fi: }
1222   }
1223 \exp_args:No \__cmd_tmp:w { \tl_to_str:n { end } }

(End of definition for \__cmd_copy_environment_end:nnNN and others.)

```

Not much to do regarding `\@latexrelease`: we could remove the entries from `\@declarecommandcopylist` but it doesn't seem worth it.

```

1224 <|latexrelease>\EndIncludeInRelease
1225 <|latexrelease>\IncludeInRelease{2020/10/01}{\__cmd_copy:NN (part 2)}%
1226 <|latexrelease> {Support~\NewCommandCopy~in~ltcmd}
1227 <|latexrelease>\EndIncludeInRelease

```

1.7.2 Showing the definition of a command

```

1228 <|latexrelease>\IncludeInRelease{2021/11/15}{\__cmd_show:N}%
1229 <|latexrelease> {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.

```

1230 \cs_new_protected:Npn \__cmd_show:N #1
1231   {

```

```

1232     \use:x
1233     {
1234         \int_set:Nn \tex_escapechar:D { 92 }
1235         \exp_not:N \__cmd_cmd_type_cases:NnnnnF \exp_not:N #1
1236             { \__cmd_show_command:N }
1237             { \__cmd_show_expandable:N }
1238             { \__cmd_show_environment:N }
1239             { \__cmd_show_environment_end:N }
1240             { \__cmd_cant_copy:nwn { non-ltcmd } }
1241             \exp_not:N #1
1242         \exp_not:N \__cmd_break_point:n { \cs_to_str:N #1 }
1243         \int_set:Nn \tex_escapechar:D { \int_use:N \tex_escapechar:D }
1244     }
1245 }
```

(End of definition for `__cmd_show:N`.)

`__cmd_show_command: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:Nnnnn`.

```

1246 \cs_new_protected:Npn \__cmd_show_command:N #1
1247     { \exp_after:wN \__cmd_show_command:Nnnnn #1 \q__cmd #1 }
1248 \cs_new_protected:Npn \__cmd_show_command:Nnnnn #1 #2 #3 #4 #5 \q__cmd #6
1249     {
1250         \__cmd_show_command_aux:Nnnnn \tl_show:x
1251         { document~command } #6 #4 {#2}
1252     }
1253 \cs_new_protected:Npn \__cmd_show_expandable:N #1
1254     { \exp_after:wN \__cmd_show_expandable:Nnnnnn #1 #1 }
1255 \end{IncludeInRelease}
1256 \IncludeInRelease{2020/10/01}{\__cmd_show:N}%
1257 \Support~\ShowCommand~in~ltcmand
1258 \EndIncludeInRelease
```

There's one variant: a command begins with `__cmd_start_expandable:nNNNNn` may still be un-expandable/protected if it's defined by `\NewDocumentCommand` and friends, with empty or only m-type arguments.

```

1259 \IncludeInRelease{2023/06/01}{\__cmd_show_expandable:Nnnnnnnn}%
1260 \Distinguish~non~expandable~document~commands
1261 \cs_new_protected:Npn \__cmd_show_expandable:Nnnnnnnn #1 #2 #3 #4 #5 #6 #7 #8
1262     {
1263         \exp_args:NNe \__cmd_show_command_aux:Nnnnn \tl_show:x
1264         { \token_if_protected_macro:NF #8 { expandable~ } document~command }
1265         #8 #5 {#2}
1266     }
1267 \EndIncludeInRelease
1268 \IncludeInRelease{2021/11/15}{\__cmd_show_expandable:Nnnnnnnn}%
1269 \Support~\ShowCommand~in~ltcmand
1270 \cs_new_protected:Npn \__cmd_show_expandable:Nnnnnnnn #1 #2 #3 #4 #5 #6 #7 #8
1271 \EndIncludeInRelease
1272 \__cmd_show_command_aux:Nnnnn \tl_show:x
1273 { expandable~document~command } #8 #5 {#2}
1274 \EndIncludeInRelease
1275 \EndIncludeInRelease
```

```

1276 <{latexrelease}>\IncludeInRelease{2020/10/01}{\_\_cmd_show_expandable:NnNNNNnN}%
1277 <{latexrelease}> {Support~\ShowCommand~in~\ltcmand}
1278 <{latexrelease}>\EndIncludeInRelease
1279 <{latexrelease}>\IncludeInRelease{2021/11/15}{\_\_cmd_show:N (part 2)}%
1280 <{latexrelease}> {Support~\ShowCommand~in~\ltcmand}

```

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:

```

1281 \cs_new_protected:Npn \_\_cmd_show_command_aux:NnNNn #1 #2 #3 #4 #5
1282 {
1283     \_\_cmd_split_signature:n {#5}
1284     #1
1285     {
1286         \token_to_str:N #3 = #2: \iow_newline:
1287         \tl_use:N \l\_\_cmd_tmpa_tl
1288         -> \cs_replacement_spec:N #4
1289     }
1290 }

```

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:NnNNn`. 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.

```

1291 \cs_new_protected:Npn \_\_cmd_show_environment:N #1
1292 {
1293     \exp_after:wN \_\_cmd_show_environment:Nnnw #1 \q\_\_cmd
1294     \tl_show:x
1295     {
1296         \token_to_str:N \end { \cs_to_str:N #1 } : \iow_newline:
1297         -> \exp_args:Nc \cs_replacement_spec:N
1298         { environment~ \cs_to_str:N #1 ~end~aux~ }
1299     }
1300 }
1301 \cs_new_protected:Npn \_\_cmd_show_environment:Nnnw #1 #2 #3 #4 \q\_\_cmd
1302 {
1303     \use:x
1304     {
1305         \_\_cmd_show_command_aux:NnNNn \_\_cmd_show:x { document~environment }
1306         { \exp_not:N \begin {#3} }
1307         \exp_not:c { environment~ #3 ~ code }
1308         {#2}
1309     }
1310 }
1311 \cs_new_protected:Npn \_\_cmd_show:x #1
1312 { \iow_term:x { > ~ #1 . \iow_newline: } }
1313 \cs_new_protected:Npn \_\_cmd_show_environment_end:N #1
1314 {
1315     \exp_args:NNx \_\_cmd_check_end:Nn \l\_\_cmd_tmpa_tl { \cs_to_str:N #1 }
1316     \exp_args:Nc \_\_cmd_show_environment:N { \l\_\_cmd_tmpa_tl }
1317 }

```

And, of course, add `__kernel_cmd_if_xparse:NTF` and `__cmd_show:N` to `\@showcommandlisthook` and to `\@showenvironmentlisthook` (`__cmd_show:N` takes care of the environment case as well, so both entries are identical):

```
1318 \tl_gput_right:Nn \@showcommandlisthook
1319   { { \__kernel_cmd_if_xparse:NTF \__cmd_show:N } }
1320 \tl_gput_right:Nn \@showenvironmentlisthook
1321   { { \__kernel_cmd_if_xparse:NTF \__cmd_show:N } }
```

(End of 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_t1` holds the final token list to be printed, and `\l__cmd_tmpb_t1` 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).

```
1322 \cs_new_protected:Npn \__cmd_split_signature:n #1
1323   {
1324     \int_set:Nn \l__cmd_current_arg_int { 1 }
1325     \tl_clear:N \l__cmd_tmpa_t1
1326     \tl_clear:N \l__cmd_tmpb_t1
1327     \__cmd_split_signature_loop:Nw #1 \q_recursion_tail \q_recursion_stop
1328   }
```

`__cmd_split_signature_loop:Nw`

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_t1`), 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.

```
1329 \cs_new_protected:Npn \__cmd_split_signature_loop:Nw #1
1330   {
1331     \quark_if_recursion_tail_stop:N #1
1332     \tl_if_empty:NT \l__cmd_tmpb_t1 { \__cmd_split_start_item: }
1333     \tl_if_exist:cTF { \c__cmd_show_type_{#1}_tl }
1334     {
1335       \use:c
1336       {
1337         \__cmd_show_
1338         \if_case:w \tl_use:c { \c__cmd_show_type_{#1}_tl } \exp_stop_f:
1339         \delim \or: delims \or: delims_opt \or: opt \or:
1340         \e \or: E \or: prefix \or: processor \fi: :Nw
1341       } #1
1342     }
1343     { \__cmd_split_end_item:n {#1} \__cmd_split_signature_loop:Nw }
1344   }
```

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;

`\c__cmd_show_type_t_tl`
`\c__cmd_show_type_r_tl`
`\c__cmd_show_type_d_tl`
`\c__cmd_show_type_R_tl`
`\c__cmd_show_type_D_tl`
`\c__cmd_show_type_0_tl`
`\c__cmd_show_type_e_tl`
`\c__cmd_show_type_E_tl`
`\c__cmd_show_type_+_tl`
`\c__cmd_show_type_!_tl`
`\c__cmd_show_type_>_tl`

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);

```

1345 \cs_set_protected:Npn \__cmd_tmp:w #1 #2
1346   {
1347     \quark_if_nil:nF {#1}
1348     { \tl_const:cn { c__cmd_show_type_#1_tl } {#2} \__cmd_tmp:w }
1349   }
1350 \__cmd_tmp:w t0 r1 d1 R2 D2 03 e4 E5 +6 !6 >7 =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.

```

1351 \cs_new_protected:Npn \__cmd_show_delim:Nw #1 #2
1352   { \__cmd_split_end_item:n { #1 #2 } \__cmd_split_signature_loop:Nw }
1353 \cs_new_protected:Npn \__cmd_show_delims:Nw #1 #2 #3
1354   { \__cmd_split_end_item:n { #1 #2 #3 } \__cmd_split_signature_loop:Nw }
1355 \cs_new_protected:Npn \__cmd_show_delims_opt:Nw #1 #2 #3 #4
1356   { \__cmd_split_end_item:n { #1 #2 #3 {#4} } \__cmd_split_signature_loop:Nw }
1357 \cs_new_protected:Npn \__cmd_show_opt:Nw #1 #2
1358   { \__cmd_split_end_item:n { #1 {#2} } \__cmd_split_signature_loop:Nw }
1359 \cs_new_protected:Npn \__cmd_show_e:Nw #1 #2
1360   {
1361     \tl_map_inline:nn {#2}
1362     {
1363       \__cmd_split_start_item:
1364       \__cmd_split_end_item:n { #1 ##1 }
1365     }
1366     \__cmd_split_signature_loop:Nw
1367   }
1368 \cs_set_protected:Npn \__cmd_tmp:w #1
1369   {
1370     \cs_new_protected:Npn \__cmd_show_E:Nw ##1 ##2 ##3
1371     {
1372       \cs_set_protected:Npn \__cmd_tmp:w #####1 #####2
1373       {
1374         \__cmd_split_start_item:
1375         \__cmd_split_end_item:n { ##1 #####1 #####2 }
1376       }
1377       \__cmd_tl_mapthread_function:nnN {##2}
1378       { ##3 {#1} {#1} {#1} {#1} {#1} {#1} {#1} {#1} {#1} } \__cmd_tmp:w
1379       \__cmd_split_signature_loop:Nw
1380     }

```

```

1381   }
1382 \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.

```

1383 \cs_new_protected:Npn \__cmd_show_prefix:Nw #1
1384   { \__cmd_split_add_item:n {#1} \__cmd_split_signature_loop:Nw }
1385 \cs_new_protected:Npn \__cmd_show_processor:Nw #1 #2
1386   { \__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.

```

1387 \cs_new_protected:Npn \__cmd_split_start_item:
1388   {
1389     \tl_set:Nx \l__cmd_tmpb_tl
1390     { ~ \c_space_tl \c_hash_str \int_use:N \l__cmd_current_arg_int : }
1391   }
1392 \cs_new_protected:Npn \__cmd_split_add_item:n #1
1393   { \tl_put_right:Nx \l__cmd_tmpb_tl { \tl_to_str:n {#1} } }
1394 \cs_new_protected:Npn \__cmd_split_end_item:n #1
1395   {
1396     \tl_put_right:Nx \l__cmd_tmpa_tl
1397     { \l__cmd_tmpb_tl \tl_to_str:n {#1} \iow_newline: }
1398     \tl_clear:N \l__cmd_tmpb_tl
1399     \int_incr:N \l__cmd_current_arg_int
1400   }

```

(End of definition for `__cmd_split_signature:n` and others.)

Not much to do regarding `\@showcommandlisthook`: we could remove the entries from `\@showcommandlisthook`, but it doesn't seem worth it.

```

\__cmd_split_start_item:
\__cmd_split_add_item:n
\__cmd_split_end_item:n
1401 <!\@release>\EndIncludeInRelease
1402 %
1403 <!\@release>\IncludeInRelease{2020/10/01}{\__cmd_show:N (part 2)}%
1404 <!\@release> {Support~\ShowCommand~in~\ltcmd}
1405 <!\@release>\EndIncludeInRelease

```

1.8 Grabbing arguments

All of the grabbers follow the same basic pattern. The initial function stores in `\l__cmd_signature_tl` the code to grab further arguments, defines (the function in) `\l__cmd_fn_tl` that will grab the argument, and calls it.

Defining `\l__cmd_fn_tl` 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, `\l__cmd_fn_tl` calls `__cmd_add_arg:n`, responsible for calling processors and grabbing further arguments.

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_tl` and also putting back the `\end` that served as an end-delimiter: this `\end` receives the environment

```

\__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

```

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.

```

1406 \cs_new_protected:Npn \__cmd_grab_b:w
1407   { \__cmd_grab_b_aux:NNw \cs_set_protected_nopar:Npn \tl_trim_spaces:n }
1408 \cs_new_protected:Npn \__cmd_grab_b_long:w
1409   { \__cmd_grab_b_aux:NNw \cs_set_protected:Npn \tl_trim_spaces:n }
1410 \cs_new_protected:Npn \__cmd_grab_b_obey_spaces:w
1411   { \__cmd_grab_b_aux:NNw \cs_set_protected_nopar:Npn \exp_not:n }
1412 \cs_new_protected:Npn \__cmd_grab_b_long_obey_spaces:w
1413   { \__cmd_grab_b_aux:NNw \cs_set_protected:Npn \exp_not:n }
1414 \cs_new_protected:Npn \__cmd_grab_b_aux:NNw #1#2#3 \__cmd_run_code:
1415   {
1416     \__cmd_grab_D_aux:NNnNN \begin \end {#3} #1 \use_i:nn
1417     \tl_put_left:Nn \l__cmd_signature_t1 { \__cmd_grab_b_end:Nw #2 }
1418     \tl_set_eq:NN \l__cmd_saved_args_t1 \l__cmd_args_t1
1419     \tl_clear:N \l__cmd_args_t1
1420     \exp_args:Nc \l__cmd_fn_t1 { begin ~ }
1421   }
1422 \cs_new_protected:Npn \__cmd_grab_b_end:Nw #1#2 \__cmd_run_code:
1423   {
1424     \tl_set:Nx \l__cmd_args_t1
1425     {
1426       \exp_not:V \l__cmd_saved_args_t1
1427       { \exp_after:wN #1 \l__cmd_args_t1 }
1428     }
1429     #2
1430     \__cmd_run_code:
1431   \end
1432 }
```

(End of definition for `__cmd_grab_b:w` and others.)

`__cmd_grab_D:w`
`__cmd_grab_D_long:w`
`__cmd_grab_D_obey_spaces:w`

The generic delimited argument grabber. The auxiliary function does a peek test before calling `__cmd_grab_D_call:Nw`, so that the optional nature of the argument works as expected.

```

1433 \cs_new_protected:Npn \__cmd_grab_D:w #1#2#3 \__cmd_run_code:
1434   {
1435     \__cmd_grab_D_aux:NNnNN #1 #2 {#3} \cs_set_protected_nopar:Npn
1436       \__cmd_peek_nonspace_remove:NTF \use_i:nn
1437   }
1438 \cs_new_protected:Npn \__cmd_grab_D_long:w #1#2#3 \__cmd_run_code:
1439   {
1440     \__cmd_grab_D_aux:NNnNN #1 #2 {#3} \cs_set_protected:Npn
1441       \__cmd_peek_nonspace_remove:NTF \use_i:nn
1442   }
1443 \cs_new_protected:Npn \__cmd_grab_D_obey_spaces:w #1#2#3 \__cmd_run_code:
1444   {
1445     \__cmd_grab_D_aux:NNnNN #1 #2 {#3} \cs_set_protected_nopar:Npn
1446       \__cmd_peek_meaning_remove:NTF \use_i:nn
1447   }
1448 \cs_new_protected:Npn \__cmd_grab_D_long_obey_spaces:w #1#2#3 \__cmd_run_code:
```

```

1449 {
1450   \__cmd_grab_D_aux:NNnNNN #1 #2 {#3} \cs_set_protected:Npn
1451     \__cmd_peek_meaning_remove:NTF \use_i:nn
1452 }
1453 \cs_new_protected:Npn \__cmd_grab_D_no_strip:w
1454   #1#2#3 \__cmd_run_code:
1455 {
1456   \__cmd_grab_D_aux:NNnNNN #1 #2 {#3} \cs_set_protected_nopar:Npn
1457     \__cmd_peek_nonspace_remove:NTF \use_none:n
1458 }
1459 \cs_new_protected:Npn \__cmd_grab_D_long_no_strip:w
1460   #1#2#3 \__cmd_run_code:
1461 {
1462   \__cmd_grab_D_aux:NNnNNN #1 #2 {#3} \cs_set_protected:Npn
1463     \__cmd_peek_nonspace_remove:NTF \use_none:n
1464 }
1465 \cs_new_protected:Npn \__cmd_grab_D_obey_spaces_no_strip:w
1466   #1#2#3 \__cmd_run_code:
1467 {
1468   \__cmd_grab_D_aux:NNnNNN #1 #2 {#3} \cs_set_protected_nopar:Npn
1469     \__cmd_peek_meaning_remove:NTF \use_none:n
1470 }
1471 \cs_new_protected:Npn \__cmd_grab_D_long_obey_spaces_no_strip:w
1472   #1#2#3 \__cmd_run_code:
1473 {
1474   \__cmd_grab_D_aux:NNnNNN #1 #2 {#3} \cs_set_protected:Npn
1475     \__cmd_peek_meaning_remove:NTF \use_none:n
1476 }

```

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.

```

1477 \cs_new_protected:Npn \__cmd_grab_D_aux:NNnNNN #1#2#3#4#5#6
1478 {
1479   \__cmd_grab_D_aux:NNnNN #1#2 {#3} #4 #6
1480   #5 #1
1481   { \__cmd_grab_D_call:Nw #1 }
1482   { \__cmd_add_arg:o \c_novalue_tl }
1483 }

```

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. To allow for suppression of brace stripping, the business end is passed here as #5.

```

1484 \cs_new_protected:Npn \__cmd_grab_D_aux:NNnNN #1#2#3#4#5
1485 {
1486   \tl_set:Nn \l__cmd_signature_tl {#3}
1487   \exp_after:wN #4 \l__cmd_fn_tl ##1 #2
1488   {
1489     \tl_if_in:nnTF {##1} {#1}
1490       { \__cmd_grab_D_nested>NNnN #1 #2 {##1} #4 }

```

```

1491     {
1492         \tl_if_blank:oTF { \use_none:n ##1 }
1493         { \__cmd_add_arg:o { \use_none:n ##1 } }
1494         {
1495             \str_if_eq:eeTF
1496             { \exp_not:o { \use_none:n ##1 } }
1497             { { \exp_not:o { \use_i:nnn ##1 \q_nil } } }
1498             { \__cmd_add_arg:o { #5 ##1 } }
1499             { \__cmd_add_arg:o { \use_none:n ##1 } }
1500         }
1501     }
1502 }
1503 }
```

(End of definition for `__cmd_grab_D:w` and others.)

`__cmd_grab_D_nested:N`
`__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.

```

1504 \tl_new:N \l__cmd_nesting_a_tl
1505 \tl_new:N \l__cmd_nesting_b_tl
1506 \quark_new:N \q__cmd
1507 \cs_new_protected:Npn \__cmd_grab_D_nested:N #1#2#3#4
1508 {
1509     \tl_clear:N \l__cmd_nesting_a_tl
1510     \tl_clear:N \l__cmd_nesting_b_tl
1511     \exp_after:wN #4 \l__cmd_fn_tl ##1 #1 ##2 \q__cmd ##3 #2
1512     {
1513         \tl_put_right:No \l__cmd_nesting_a_tl { \use_none:n ##1 #1 }
1514         \tl_put_right:No \l__cmd_nesting_b_tl { \use_i:nn #2 ##3 }
1515         \tl_if_in:nnTF {##2} {#1}
1516         {
1517             \l__cmd_fn_tl
1518             \q_nil ##2 \q__cmd \ERROR
1519         }
1520     {
1521         \tl_put_right:Nx \l__cmd_nesting_a_tl
1522             { \__cmd_grab_D_nested:w \q_nil ##2 \q_stop }
1523         \tl_if_in:NnTF \l__cmd_nesting_b_tl {#1}
1524         {
1525             \tl_set_eq:NN \l__cmd_tmpa_tl \l__cmd_nesting_b_tl
1526             \tl_clear:N \l__cmd_nesting_b_tl
1527             \exp_after:wN \l__cmd_fn_tl \exp_after:wN
1528                 \q_nil \l__cmd_tmpa_tl \q_nil \q__cmd \ERROR
```

```

1529         }
1530     {
1531         \tl_put_right:No \l__cmd_nesting_a_tl
1532             \l__cmd_nesting_b_tl
1533             \__cmd_add_arg:V \l__cmd_nesting_a_tl
1534     }
1535 }
1536 \l__cmd_fn_tl #3 \q_nil \q_cmd \ERROR
1537 }
1538 \cs_new:Npn \__cmd_grab_D_nested:w #1 \q_nil \q_stop
1539 { \exp_not:o { \use_none:n #1 } }

```

(End of 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.

```

1541 \cs_new_protected_nopar:Npn \__cmd_grab_D_call:Nw #1
1542 {
1543     \token_if_eq_catcode:NNTF + #1
1544     {
1545         \exp_after:wN \exp_after:wN \exp_after:wN
1546             \l__cmd_fn_tl \char_generate:nn { '#1 } { 11 }
1547     }
1548     {
1549         \__cmd_token_if_cs:NTF #1
1550         {
1551             \exp_after:wN \l__cmd_fn_tl
1552                 \cs:w \cs_to_str:N #1 ~ \cs_end:
1553         }
1554         {
1555             \exp_after:wN \l__cmd_fn_tl
1556                 \token_to_str:N #1
1557         }
1558     }
1559 }

```

(End of definition for `__cmd_grab_D_call:Nw`.)

`__cmd_grab_E:w` Everything here needs to point to a loop.

```

\__cmd_grab_E_long:w
\__cmd_grab_E_long:W
\__cmd_grab_E_obey_spaces:w
\cmd_grab_E_long_obey_spaces:w
\__cmd_grab_E:nnNN
\__cmd_grab_E_loop:NnN
\__cmd_grab_E_finalise:

```

File g: ltcmd.dtx Date: 2023-05-26 Version v1.1e

```

1561   {
1562     \__cmd_grab_E:nNn {#1} {#2}
1563     \cs_set_protected_nopar:Npn
1564     \__cmd_peek_nonspace_remove:NTF
1565   }
1566 \cs_new_protected:Npn \__cmd_grab_E_long:w #1#2 \__cmd_run_code:
1567   {
1568     \__cmd_grab_E:nNn {#1} {#2}
1569     \cs_set_protected:Npn
1570     \__cmd_peek_nonspace_remove:NTF
1571   }
1572 \cs_new_protected:Npn \__cmd_grab_E_obey_spaces:w #1#2 \__cmd_run_code:
1573   {
1574     \__cmd_grab_E:nNn {#1} {#2}
1575     \cs_set_protected_nopar:Npn
1576     \__cmd_peek_meaning_remove:NTF
1577   }
1578 \cs_new_protected:Npn \__cmd_grab_E_long_obey_spaces:w #1#2 \__cmd_run_code:
1579   {
1580     \__cmd_grab_E:nNn {#1} {#2}
1581     \cs_set_protected:Npn
1582     \__cmd_peek_meaning_remove:NTF
1583   }

```

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.

```

1584 \cs_new_protected:Npn \__cmd_grab_E:nNn #1#2#3#4
1585   {
1586     \exp_after:wN #3 \l__cmd_fn_tl ##1##2##3
1587     {
1588       \prop_put:Nnn \l__cmd_tmp_prop {##1} {##3}
1589       \__cmd_grab_E_loop:NnN #4 { } ##2 \q_recursion_stop
1590     }
1591     \prop_clear:N \l__cmd_tmp_prop
1592     \tl_set:Nn \l__cmd_signature_tl {#2}
1593     \cs_set_protected:Npn \__cmd_grab_E_finalise:
1594     {
1595       \tl_map_inline:nn {#1}
1596       {
1597         \prop_get:NnNF \l__cmd_tmp_prop {####1} \l__cmd_tmpb_tl
1598         { \tl_set_eq:NN \l__cmd_tmpb_tl \c_no_value_tl }
1599         \tl_put_right:Nx \l__cmd_args_tl
1600         { { \exp_not:V \l__cmd_tmpb_tl } }
1601       }
1602       \l__cmd_signature_tl \__cmd_run_code:
1603     }
1604     \__cmd_grab_E_loop:NnN #4 { } #1 \q_recursion_tail \q_recursion_stop
1605   }
1606 \cs_new_protected:Npn \__cmd_grab_E_loop:NnN #1#2#3#4 \q_recursion_stop
1607   {
1608     \cs_if_eq:NNTF #3 \q_recursion_tail
1609     { \__cmd_grab_E_finalise: }

```

```

1610      {
1611          #1 #3
1612          { \l__cmd_fn_tl #3 {#2#4} }
1613          { \__cmd_grab_E_loop:NnN #1 {#2#3} #4 \q_recursion_stop }
1614      }
1615  }
1616 \cs_new_protected:Npn \__cmd_grab_E_finalise: { }

(End of definition for \__cmd_grab_E:w and others.)

```

Collecting a single mandatory argument is quite easy.

```

\__cmd_grab_m:w
\__cmd_grab_m_long:w
1617 \cs_new_protected:Npn \__cmd_grab_m:w #1 \__cmd_run_code:
1618  {
1619      \tl_set:Nn \l__cmd_signature_tl {#1}
1620      \exp_after:wN \cs_set_protected_nopar:Npn \l__cmd_fn_tl ##1
1621          { \__cmd_add_arg:n {##1} }
1622      \l__cmd_fn_tl
1623  }
1624 \cs_new_protected:Npn \__cmd_grab_m_long:w #1 \__cmd_run_code:
1625  {
1626      \tl_set:Nn \l__cmd_signature_tl {#1}
1627      \exp_after:wN \cs_set_protected:Npn \l__cmd_fn_tl ##1
1628          { \__cmd_add_arg:n {##1} }
1629      \l__cmd_fn_tl
1630  }

```

(End of 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

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_aux:Nnnnnnnnn
1631 \cs_new_protected_nopar:Npn \__cmd_grab_m_aux:Nnnnnnnnn #1#2#3#4#5#6#7#8#9
1632  {
1633      \tl_put_right:No \l__cmd_args_tl
1634          { #1 {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9} }
1635      \l__cmd_signature_tl \__cmd_run_code:
1636  }
1637 \cs_new_protected:cpn { __cmd_grab_m_1:w } #1 \__cmd_run_code:
1638  {
1639      \tl_set:Nn \l__cmd_signature_tl {#1}
1640      \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
1641          \l__cmd_fn_tl \use_none:nnnnnn { } { } { } { } { } { }
1642  }
1643 \cs_new_protected:cpn { __cmd_grab_m_2:w } #1 \__cmd_run_code:
1644  {
1645      \tl_set:Nn \l__cmd_signature_tl {#1}
1646      \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
1647          \l__cmd_fn_tl \use_none:nnnnnn { } { } { } { } { } { }
1648  }
1649 \cs_new_protected:cpn { __cmd_grab_m_3:w } #1 \__cmd_run_code:
1650  {
1651      \tl_set:Nn \l__cmd_signature_tl {#1}
1652      \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
1653          \l__cmd_fn_tl \use_none:nnnnnn { } { } { } { } { }

```

```

1654      }
1655  \cs_new_protected:cpn { __cmd_grab_m_4:w } #1 \__cmd_run_code:
1656  {
1657      \tl_set:Nn \l__cmd_signature_tl {#1}
1658      \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
1659      \l__cmd_fn_tl \use_none:n { } { } { } { }
1660  }
1661  \cs_new_protected:cpn { __cmd_grab_m_5:w } #1 \__cmd_run_code:
1662  {
1663      \tl_set:Nn \l__cmd_signature_tl {#1}
1664      \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
1665      \l__cmd_fn_tl \use_none:n { } { } { }
1666  }
1667  \cs_new_protected:cpn { __cmd_grab_m_6:w } #1 \__cmd_run_code:
1668  {
1669      \tl_set:Nn \l__cmd_signature_tl {#1}
1670      \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
1671      \l__cmd_fn_tl \use_none:nn { } { }
1672  }
1673  \cs_new_protected:cpn { __cmd_grab_m_7:w } #1 \__cmd_run_code:
1674  {
1675      \tl_set:Nn \l__cmd_signature_tl {#1}
1676      \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
1677      \l__cmd_fn_tl \use_none:n { }
1678  }
1679  \cs_new_protected:cpn { __cmd_grab_m_8:w } #1 \__cmd_run_code:
1680  {
1681      \tl_set:Nn \l__cmd_signature_tl {#1}
1682      \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
1683      \l__cmd_fn_tl \prg_do_nothing:
1684  }
1685  \cs_new_protected:cpn { __cmd_grab_m_9:w }
1686  {
1687      \exp_not:c { __cmd_grab_m_5:w }
1688      \exp_not:c { __cmd_grab_m_4:w }
1689  }

```

(End of definition for `__cmd_grab_m_1:w` and others.)

`__cmd_grab_R:w`
`__cmd_grab_R_long:w`
`__cmd_grab_R_aux:NnnN`

```

1690  \cs_new_protected:Npn \__cmd_grab_R:w #1#2#3 \__cmd_run_code:
1691  {
1692      \__cmd_grab_R_aux:NNnN #1 #2 {#3} \cs_set_protected_nopar:Npn
1693  \cs_new_protected:Npn \__cmd_grab_R_long:w #1#2#3 \__cmd_run_code:
1694  {
1695      \__cmd_grab_R_aux:NNnN #1 #2 {#3} #4 \use_ii:nn
1696      \__cmd_peek_nonspace_remove:NTF #1
1697      {
1698          \__cmd_grab_D_call:Nw #1
1699          {
1700              \msg_error:nnxx { cmd } { missing-required }
1701              {
1702                  \__cmd_environment_or_command:
1703                  \token_to_str:N #1

```

```

1703           \__cmd_add_arg:o \c_novalue_tl
1704       }
1705   }

```

(End of 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.

```

1706 \cs_new_protected:Npn \__cmd_grab_t:w
1707   { \__cmd_grab_t_aux:NNw \__cmd_peek_nonspace_remove:NTF }
1708 \cs_new_protected:Npn \__cmd_grab_t_obey_spaces:w
1709   { \__cmd_grab_t_aux:NNw \__cmd_peek_meaning_remove:NTF }
1710 \cs_new_protected:Npn \__cmd_grab_t_aux:NNw #1#2#3 \__cmd_run_code:
1711   {
1712     \tl_set:Nn \l__cmd_signature_tl {#3}
1713     \exp_after:wN \cs_set_protected:Npn \l__cmd_fn_tl
1714     {
1715       #1 #2
1716       { \__cmd_add_arg:n { \BooleanTrue } }
1717       { \__cmd_add_arg:n { \BooleanFalse } }
1718     }
1719     \l__cmd_fn_tl
1720   }

```

(End of definition for `__cmd_grab_t:w`, `__cmd_grab_t_obey_spaces:w`, and `__cmd_grab_t_aux:NNw`.)

\l__cmd_v_arg_tl

`__cmd_grab_v:w` Firstly, it is necessary to change `\tex_endlinechar:D` so that newlines in different catcode regimes (e.g., `\ExplSyntaxOn`) are not misinterpreted as spaces. 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.

```

1722 \cs_new_protected:Npn \__cmd_grab_v:w
1723   {
1724     \bool_set_false:N \l__cmd_long_bool
1725     \__cmd_grab_v_aux:w
1726   }
1727 \cs_new_protected:Npn \__cmd_grab_v_long:w
1728   {
1729     \bool_set_true:N \l__cmd_long_bool
1730     \__cmd_grab_v_aux:w

```

```

1731   }
1732 \cs_new_protected:Npn \__cmd_grab_v_aux:w #1 \__cmd_run_code:
1733 {
1734   \tl_set:Nn \l__cmd_signature_tl {\#1}
1735   \group_begin:
1736     \tex_escapechar:D = 92 \scan_stop:
1737     \tex_endlinechar:D = '\^M \scan_stop:
1738     \tl_clear:N \l__cmd_v_arg_tl
1739     \peek_remove_spaces:n
1740   {
1741     \peek_meaning_remove:NTF \c_group_begin_token
1742     {
1743       \group_align_safe_end:
1744       \__cmd_grab_v_bgroup:
1745     }
1746     {
1747       \peek_N_type:TF
1748         { \__cmd_grab_v_aux_test:N }
1749         { \__cmd_grab_v_aux_abort:n { } }
1750     }
1751   }
1752 }
1753 \cs_new_protected:Npn \__cmd_grab_v_group_end:
1754 {
1755   \exp_args:NNNo
1756   \group_end:
1757   \tl_set:Nn \l__cmd_v_arg_t1 { \l__cmd_v_arg_t1 }
1758 }

```

(End of 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_t1`. 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.

```

1759 \cs_new_protected:Npn \__cmd_grab_v_aux_test:N #1
1760 {
1761   \__cmd_grab_v_token_if_char:NTF #1
1762   {
1763     \__cmd_grab_v_aux_put:N #1
1764     \__cmd_grab_v_aux_catcodes:
1765     \__cmd_grab_v_aux_loop:N #1
1766   }
1767   { \__cmd_grab_v_aux_abort:n {\#1} #1 }
1768 }
1769 \cs_new_protected:Npn \__cmd_grab_v_aux_loop:N #1

```

```

1770   {
1771     \peek_N_type:TF
1772     { \__cmd_grab_v_aux_loop:NN #1 }
1773     { \__cmd_grab_v_aux_abort:n { } }
1774   }
1775 \cs_new_protected:Npn \__cmd_grab_v_aux_loop:NN #1#2
1776   {
1777     \__cmd_grab_v_token_if_char:NTF #2
1778     {
1779       \token_if_eq_charcode:NNTF #1 #2
1780       { \__cmd_grab_v_aux_loop_end: }
1781       {
1782         \__cmd_grab_v_aux_put:N #2
1783         \__cmd_grab_v_aux_loop:N #1
1784       }
1785     }
1786     { \__cmd_grab_v_aux_abort:n {#2} #2 }
1787   }
1788 \cs_new_protected:Npn \__cmd_grab_v_aux_loop_end:
1789   {
1790     \__cmd_grab_v_group_end:
1791     \__cmd_add_arg:x { \tl_tail:N \l__cmd_v_arg_tl }
1792   }

```

(End of definition for `__cmd_grab_v_aux_test:N` and others.)

`\l__cmd_v_nesting_int`

`\int_new:N \l__cmd_v_nesting_int`

`__cmd_grab_v_bgroup:` If the opening delimiter is a left brace, we keep track of how many left and right braces were encountered so far in `\l__cmd_v_nesting_int` (the methods used for optional arguments cannot apply here), and stop as soon as it reaches 0.

Some care was needed when removing the opening delimiter, which has already been assigned category code 1: using `\peek_meaning_remove:NTF` in the `__cmd_grab_v_aux:w` function would break within alignments. Instead, we first convert that token to a string, and remove the result as a normal undelimited argument.

```

1794 \cs_new_protected:Npx \__cmd_grab_v_bgroup:
1795   {
1796     \exp_not:N \__cmd_grab_v_aux_catcodes:
1797     \exp_not:n { \int_set:Nn \l__cmd_v_nesting_int { 1 } }
1798     \exp_not:N \__cmd_grab_v_aux_put:N \iow_char:N \{
1799     \exp_not:N \__cmd_grab_v_bgroup_loop:
1800   }
1801 \cs_new_protected:Npn \__cmd_grab_v_bgroup_loop:
1802   {
1803     \peek_N_type:TF
1804     { \__cmd_grab_v_bgroup_loop:N }
1805     { \__cmd_grab_v_aux_abort:n { } }
1806   }
1807 \cs_new_protected:Npn \__cmd_grab_v_bgroup_loop:N #1
1808   {
1809     \__cmd_grab_v_token_if_char:NTF #1
1810     {

```

```

1811     \token_if_eq_charcode:NNTF \c_group_end_token #1
1812     {
1813         \int_decr:N \l__cmd_v_nesting_int
1814         \int_compare:nNnTF \l__cmd_v_nesting_int > 0
1815         {
1816             \__cmd_grab_v_aux_put:N #1
1817             \__cmd_grab_v_bgroup_loop:
1818         }
1819         { \__cmd_grab_v_aux_loop_end: }
1820     }
1821     {
1822         \token_if_eq_charcode:NNT \c_group_begin_token #1
1823         { \int_incr:N \l__cmd_v_nesting_int }
1824         \__cmd_grab_v_aux_put:N #1
1825         \__cmd_grab_v_bgroup_loop:
1826     }
1827 }
1828 { \__cmd_grab_v_aux_abort:n {#1} #1 }
1829 }
```

(End of 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.

```

1830 \cs_new_protected:Npn \__cmd_grab_v_aux_catcodes:
1831 {
1832     \cs_set_eq:NN \do \char_set_catcode_other:N
1833     \dospecials
1834     \bool_if:NTF \l__cmd_long_bool
1835     { \char_set_catcode_other:n { \tex_endlinechar:D } }
1836     { \char_set_catcode_parameter:n { \tex_endlinechar:D } }
1837 }
1838 \cs_new_protected:Npn \__cmd_grab_v_aux_abort:n #1
1839 {
1840     \__cmd_grab_v_group_end:
1841     \exp_after:wN \exp_after:wN \exp_after:wN
1842     \peek_meaning_remove:NTF \char_generate:nn { \tex_endlinechar:D } { 6 }
1843     {
1844         \msg_error:nnnn { cmd } { verbatim-nl }
1845         { \__cmd_environment_or_command: }
1846         { \tl_to_str:N \l__cmd_v_arg_tl }
1847         { \tl_to_str:n {#1} }
1848         \__cmd_add_arg:o \c_novalue_tl
1849     }
1850     {
1851         \msg_error:nnnn { cmd } { verbatim-tokenized }
1852         { \__cmd_environment_or_command: }
1853         { \tl_to_str:N \l__cmd_v_arg_tl }
1854         { \tl_to_str:n {#1} }
1855         \__cmd_add_arg:o \c_novalue_tl
1856     }
1857 }
```

(End of 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).

```
1858 \cs_new_protected:Npn \_cmd_grab_v_aux_put:N #1
1859   {
1860     \tl_put_right:Nx \l__cmd_v_arg_tl
1861     {
1862       \token_if_active:NTF #1
1863       { \exp_not:N #1 } { \token_to_str:N #1 }
1864     }
1865   }
```

(End of 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.

```
1866 \cs_new_protected:Npn \_cmd_grab_v_token_if_char:NTF #1
1867   { \str_if_eq:eeTF { } { \str_tail:n {#1} } }
```

(End of 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`.

```
1868 \cs_new_protected:Npn \_cmd_add_arg:n #1
1869   {
1870     \tl_put_right:Nn \l__cmd_args_tl { {#1} }
1871     \l__cmd_signature_tl \_cmd_run_code:
1872   }
1873 \cs_generate_variant:Nn \_cmd_add_arg:n { V , o , x }
```

(End of 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 `\(function)_w` and `\(function)_u` are just after `\q_cmd`, we go and find them (and use the long one).

```
1874 \cs_new:Npn \_cmd_expandable_grab_D:w #1 \q_cmd #2#3
1875   { #2 { \_cmd_expandable_grab_D:NNNwNNNn #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 `\par` (because a later mandatory

²It is obvious that if #7 matches #2 then the strings are equal. We must check the converse. The right-hand-side of `\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 `\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.

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 `\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 `__cmd_tmp:w` allows to insert `-NoValue-` expanded.

```
1876 \cs_set_protected:Npn \__cmd_tmp:w #1
1877 {
1878   \cs_new:Npn \__cmd_expandable_grab_D:NNNwNNn ##1##2##3##4 \q__cmd ##5##6##7
1879   {
1880     \str_if_eq:nnTF {##2} {##7}
1881     {
1882       \str_if_eq:onTF
1883         { ##1 { } { } ##7 ##2 \q__cmd ##3 }
1884         { { } {##2} { } }
1885     }
1886     { \use_ii:nn }
1887     {
1888       ##1
1889       { \__cmd_expandable_grab_D:NNNwNNnn ##1##2##3##4 \q__cmd ##5##6 }
1890       \q_nil { } ##2 \ERROR \q__cmd \ERROR
1891     }
1892     { ##4 {#1} \q__cmd ##5 ##6 {##7} }
1893   }
1894 }
1895 \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.

```
1896 \cs_new:Npn \__cmd_expandable_grab_D:NNNwNNnn ##1##2##3##4 \q__cmd ##5##6##7##8##9
1897 {
1898   \exp_args:Nof \__cmd_expandable_grab_D:nnNNNwNN
1899   { \use_ii:nn #7 #2 }
1900   { \__cmd_expandable_grab_D:Nw #3 \exp_stop_f: #8 #9 }
1901   #1##2##3 ##4 \q__cmd #5 #6
1902 }
```

```

1903 \cs_new:Npn \__cmd_expandable_grab_D:Nw #1#2 \ERROR \ERROR { #2 #1 }

1904 \cs_new:Npn \__cmd_expandable_grab_D:nnNNNwNN #1#2#3#4#5#6 \q__cmd #7#8
1905 {
1906   \exp_args:No \tl_if_empty:oTF
1907   { #3 { \use_none:nnn } #2 \q__cmd #5 #4 \q__cmd #5 }
1908   {
1909     \tl_if_blank:oTF { \use_none:nn #1#2 }
1910     { \__cmd_put_arg_expandable:ow { \use_none:nn #1#2 } }
1911     {
1912       \str_if_eq:eeTF
1913       { \exp_not:o { \use_none:nn #1#2 } }
1914       { { \exp_not:o { \use_iii:nnnn #1#2 \q_nil } } }
1915       { \__cmd_put_arg_expandable:ow { \use_iii:nnn #1#2 } }
1916       { \__cmd_put_arg_expandable:ow { \use_none:nn #1#2 } }
1917     }
1918     #6 \q__cmd #7 #8
1919   }
1920   {
1921     #3
1922     { \__cmd_expandable_grab_D:NNNwNNnnn #3#4#5#6 \q__cmd #7 #8 }
1923     \q_nil {#1} #2 \ERROR \q__cmd \ERROR
1924   }
1925 }

```

(End of 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`

```

1926 \cs_new:Npn \__cmd_expandable_grab_D_alt:w #1 \q__cmd #2#3
1927   { #2 { \__cmd_expandable_grab_D_alt>NNwNNn #1 \q__cmd #2 #3 } }
1928 \cs_set_protected:Npn \__cmd_tmp:w #
1929   {
1930     \cs_new:Npn \__cmd_expandable_grab_D_alt>NNwNNn ##1##2##3 \q__cmd ##4##5##6
1931     {
1932       \str_if_eq:nnTF {##6} {##2}
1933       {
1934         \str_if_eq:onTF
1935         { ##1 { } ##6 ##2 ##2 }
1936         { { } ##2 }
1937       }
1938       { \use_ii:nn }
1939     }
1940     ##1
1941     { \__cmd_expandable_grab_D_alt>NNwn ##4 ##5 ##3 \q__cmd }

```

```

1942          ##6 \ERROR
1943      }
1944      { ##3 {#1} \q_cmd ##4 ##5 {##6} }
1945  }
1946  }
1947 \exp_args:No \_cmd_tmp:w { \c_novalue_t1 }
1948 \cs_new:Npn \_cmd_expandable_grab_D_alt:NNwn #1#2#3 \q_cmd #4
1949 {
1950     \tl_if_blank:oTF { \use_none:n #4 }
1951     { \_cmd_put_arg_expandable:ow { \use_none:n #4 } }
1952     {
1953         \str_if_eq:eeTF
1954         { \exp_not:o { \use_none:n #4 } }
1955         { \exp_not:o { \use_i:nnn #4 \q_nil } }
1956         { \_cmd_put_arg_expandable:ow { \use_i:nn #4 } }
1957         { \_cmd_put_arg_expandable:ow { \use_none:n #4 } }
1958     }
1959     #3 \q_cmd #1 #2
1960 }

```

(End of 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 `{-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.

```

1961 \cs_new:Npn \_cmd_expandable_grab_E:w #1 \q_cmd #2#3
1962   { \_cmd_expandable_grab_E_aux:w #1 \q_cmd #2 #3 #3 }
1963 \cs_new:Npn \_cmd_expandable_grab_E_long:w #1 \q_cmd #2#3
1964   { \_cmd_expandable_grab_E_aux:w #1 \q_cmd #2 #3 #2 }
1965 \cs_new:Npn \_cmd_expandable_grab_E_aux:w #1 \q_cmd #2#3#4
1966   { #2 { \_cmd_expandable_grab_E_test:nnw #1 \q_cmd #2 #3 #4 } }
1967 \cs_new:Npn \_cmd_expandable_grab_E_test:nnw #1#2#3 \q_cmd #4#5#6#7
1968   {
1969     \_cmd_expandable_grab_E_loop:nnnNNw {#7} { } { }
1970     #1 \q_nil \q_nil \q_nil \q_mark #2 \q_nil
1971     #3 \q_cmd #4 #5 #6
1972   }
1973 \cs_new:Npn \_cmd_expandable_grab_E_loop:nnnNNw

```

```

1974      #1#2#3#4#5#6 \q_nil #7 \q_mark #
1975      {
1976          \quark_if_nil:NTF #4
1977          { \__cmd_expandable_grab_E_end:nnw {#1} {#3} }
1978          {
1979              \t1_if_novalue:nTF {#8}
1980              { \str_if_eq:onTF { #4 { } #1 #5 } {#5} }
1981              { \use_ii:nn }
1982              { \__cmd_expandable_grab_E_find:w { #2 #4 #5 #6 } {#3} ~ }
1983              {
1984                  \__cmd_expandable_grab_E_loop:nnnNNw
1985                  {#1} { #2 #4 #5 } { #3 {#8} }
1986                  #6 \q_nil #7 \q_mark
1987              }
1988          }
1989      }
1990      \cs_new:Npn \__cmd_expandable_grab_E_find:w #1 \q_cmd #2#3#4
1991      { #4 { \__cmd_expandable_grab_E_find:nnw #1 \q_cmd #2 #3 #4 } }
1992      \cs_new:Npn \__cmd_expandable_grab_E_find:nnw #1#2#3 \q_nil #4 \q_cmd #5#6#7#8
1993      { \__cmd_expandable_grab_E_aux:w {#1} { #2 {#8} #3 } #4 \q_cmd #5 #6 #7 }
1994      \cs_new:Npn \__cmd_expandable_grab_E_end:nnw #1#2#3 \q_cmd #4#5#6
1995      { #3 #2 \q_cmd #4 #5 {#1} }

```

(End of definition for `__cmd_expandable_grab_E:w` and others.)

```
\__cmd_expandable_grab_m:w
\__cmd_expandable_grab_m_long:w
\__cmd_expandable_grab_m_aux:wNn
```

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`.

```

1996 \cs_new:Npn \__cmd_expandable_grab_m:w #1 \q_cmd #2#3
1997  { #3 { \__cmd_expandable_grab_m_aux:wNn #1 \q_cmd #2 #3 } }
1998 \cs_new:Npn \__cmd_expandable_grab_m_long:w #1 \q_cmd #2#3
1999  { #2 { \__cmd_expandable_grab_m_aux:wNn #1 \q_cmd #2 #3 } }
2000 \cs_new:Npn \__cmd_expandable_grab_m_aux:wNn #1 \q_cmd #2#3#4
2001  { #1 {#4} \q_cmd #2 #3 }
```

(End of 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
\__cmd_expandable_grab_R_aux>NNNwNNn
```

Much the same as for the D-type argument, with only the lead-off function varying.

```

2002 \cs_new:Npn \__cmd_expandable_grab_R:w #1 \q_cmd #2#3
2003  { #2 { \__cmd_expandable_grab_R_aux>NNNwNNn #1 \q_cmd #2#3 } }
2004 \cs_set_protected:Npn \__cmd_tmp:w #1
2005  {
2006      \cs_new:Npn \__cmd_expandable_grab_R_aux>NNNwNNn ##1##2##3##4 \q_cmd ##5##6##7
2007      {
2008          \str_if_eq:nnTF {##7} {##2}
2009          {
2010              \str_if_eq:onTF
2011              { ##1 { } { } ##7 ##2 \q_cmd ##3 }
2012              { { } {##2} { } }
2013          }
2014          { \use_ii:nn }
2015          {
2016              ##1
2017              { \__cmd_expandable_grab_D>NNNwNNnn ##1##2##3##4 \q_cmd ##5##6 }
2018              \q_nil { } ##2 \ERROR \q_cmd \ERROR }
```

```

2019     }
2020     {
2021         \msg_expandable_error:n { cmd } { missing-required }
2022         { \exp_args:Nf \tl_trim_spaces:n { \token_to_str:N ##5 } }
2023         { \tl_to_str:n {##2} }
2024         ##4 {#1} \q_cmd ##5 ##6 {##7}
2025     }
2026 }
2027 }
2028 \exp_args:No \__cmd_tmp:w { \c_novalue_tl }

(End of definition for \__cmd_expandable_grab_R:w and \__cmd_expandable_grab_R_aux:NNwNNn.)

```

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.

```

2029 \cs_new:Npn \__cmd_expandable_grab_R_alt:w #1 \q_cmd #2#3
2030     { #2 { \__cmd_expandable_grab_R_alt_aux:NNwNNn #1 \q_cmd #2#3 } }
2031 \cs_set_protected:Npn \__cmd_tmp:w #1
2032     {
2033         \cs_new:Npn \__cmd_expandable_grab_R_alt_aux:NNwNNn ##1##2##3 \q_cmd ##4##5##6
2034         {
2035             \str_if_eq:nnTF {##6} {##2}
2036             {
2037                 \str_if_eq:onTF
2038                     { ##1 { } ##6 ##2 ##2 }
2039                     { { } ##2 }
2040             }
2041             { \use_ii:nn }
2042             {
2043                 ##1
2044                 { \__cmd_expandable_grab_D_alt>NNwn ##4 ##5 ##3 \q_cmd }
2045                 ##6 \ERROR
2046             }
2047             {
2048                 \msg_expandable_error:n { cmd } { missing-required }
2049                 { \exp_args:Nf \tl_trim_spaces:n { \token_to_str:N ##4 } }
2050                 { \tl_to_str:n {##2} }
2051                 ##3 {#1} \q_cmd ##4 ##5 {##6}
2052             }
2053         }
2054     }
2055 \exp_args:No \__cmd_tmp:w { \c_novalue_tl }

(End of 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.

```

2056 \cs_new:Npn \__cmd_expandable_grab_t:w #1 \q_cmd #2#3
2057     { #2 { \__cmd_expandable_grab_t_aux>NNwn #1 \q_cmd #2 #3 } }
2058 \cs_new:Npn \__cmd_expandable_grab_t_aux>NNwn #1#2#3 \q_cmd #4#5#6
2059     {
2060         \str_if_eq:onTF { #1 { } #6 #2 } {##2}
2061         { #3 { \BooleanTrue } \q_cmd #4 #5 }

```

```
2062     { #3 { \BooleanFalse } \q__cmd #4 #5 {#6} }
2063 }
```

(End of definition for `__cmd_expandable_grab_t:w` and `__cmd_expandable_grab_t_aux:Nnwn`.)

`__cmd_put_arg_expandable:nw` A useful helper, to store arguments when they are ready.

```
2064 \cs_new:Npn \__cmd_put_arg_expandable:nw #1#2 \q__cmd { #2 {#1} \q__cmd }
2065 \cs_generate_variant:Nn \__cmd_put_arg_expandable:nw { o }
```

(End of definition for `__cmd_put_arg_expandable:nw`.)

1.10 Argument processors

`__cmd_bool_reverse:N` A simple reversal.

```
2066 \cs_new_protected:Npn \__cmd_bool_reverse:N #1
2067 {
2068     \bool_if:NTF #1
2069     { \tl_set:Nn \ProcessedArgument { \c_false_bool } }
2070     { \tl_set:Nn \ProcessedArgument { \c_true_bool } }
2071 }
```

(End of definition for `__cmd_bool_reverse:N`.)

```
\l__cmd_split_list_seq
```

```
\l__cmd_split_list_tl
```

Splitting can take place either at a single token or at a longer identifier. To deal with single active tokens, a two-part procedure is needed.

```
2072 \seq_new:N \l__cmd_split_list_seq
2073 \tl_new:N \l__cmd_split_list_tl
2074 \cs_new_protected:Npn \__cmd_split_list:nn #1#2
2075 {
2076     \tl_if_single:nTF {#1}
2077     {
2078         \token_if_cs:NTF #1
2079         { \__cmd_split_list_multi:nn {#1} {#2} }
2080         { \__cmd_split_list_single:Nn #1 {#2} }
2081     }
2082     { \__cmd_split_list_multi:nn {#1} {#2} }
2083 }
2084 \cs_new_protected:Npn \__cmd_split_list_multi:nn #1#2
2085 {
2086     \seq_set_split:Nnn \l__cmd_split_list_seq {#1} {#2}
2087     \tl_clear:N \ProcessedArgument
2088     \seq_map_inline:Nn \l__cmd_split_list_seq
2089     { \tl_put_right:Nn \ProcessedArgument { {##1} } }
2090 }
2091 \cs_generate_variant:Nn \__cmd_split_list_multi:nn { nV }
2092 \group_begin:
2093 \char_set_catcode_active:N \^^@
2094 \cs_new_protected:Npn \__cmd_split_list_single:Nn #1#2
2095 {
2096     \tl_set:Nn \l__cmd_split_list_tl {#2}
2097     \group_begin:
2098     \char_set_lccode:nn { \^^@ } { '#1 }
2099     \tex_lowercase:D
2100     {
2101         \group_end:
2102         \tl_replace_all:Nnn \l__cmd_split_list_tl { ^@ }
2103         {#1}
2104         \__cmd_split_list_multi:nV {#1} \l__cmd_split_list_tl
2105     }
2106 }
```

```
\group_end:
```

(End of 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
```

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.

```
2107 \cs_new_protected:Npn \__cmd_split_argument:nnn #1#2#3
2108 {
2109     \__cmd_split_list:nn {#2} {#3}
2110     \exp_args:Nf \__cmd_split_argument_aux:nnnn
2111     { \tl_count:N \ProcessedArgument }
2112     {#1} {#2} {#3}
2113 }
```

```

2114 \cs_new_protected:Npn \__cmd_split_argument_aux:nnnn #1#2#3#4
2115   {
2116     \int_compare:nNnF {#1} = { #2 + 1 }
2117     {
2118       \int_compare:nNnTF {#1} > { #2 + 1 }
2119       {
2120         \tl_set:Nx \ProcessedArgument
2121         {
2122           \exp_last_unbraced:NnNo
2123             \__cmd_split_argument_aux:n
2124             { #2 + 1 }
2125             \use_none_delimit_by_q_stop:w
2126             \ProcessedArgument
2127             \q_stop
2128         }
2129         \msg_error:nxxxx { cmd } { arg-split }
2130         { \tl_to_str:n {#3} } { \int_eval:n { #2 + 1 } }
2131         { \tl_to_str:n {#4} }
2132       }
2133     {
2134       \tl_put_right:Nx \ProcessedArgument
2135       {
2136         \prg_replicate:nn { #2 + 1 - (#1) }
2137         { { \exp_not:V \c_novalue_tl } }
2138       }
2139     }
2140   }
2141 }
```

Auxiliaries to leave exactly the correct number of arguments in \ProcessedArgument.

```

2142 \cs_new:Npn \__cmd_split_argument_aux:n #1
2143   { \prg_replicate:nn {#1} { \__cmd_split_argument_aux:wn } }
2144 \cs_new:Npn \__cmd_split_argument_aux:wn #1 \use_none_delimit_by_q_stop:w #2
2145   {
2146     \exp_not:n { {#2} }
2147     #1
2148     \use_none_delimit_by_q_stop:w
2149   }
```

(End of definition for __cmd_split_argument:nnn and others.)

__cmd_trim_spaces:n This one is almost trivial.

```

2150 \cs_new_protected:Npn \__cmd_trim_spaces:n #1
2151   { \tl_set:Nx \ProcessedArgument { \tl_trim_spaces:n {#1} } }
```

(End of definition for __cmd_trim_spaces:n.)

1.11 Conversion to key–value form

This is implemented as a process but with no public interfaces, hence is treated separately from the others: it's a feature of \tcmd which just happens to use the same mechanism as a processor.

The two clear-cut cases have been eliminated, and we therefore have to deal with a search for = signs. We need an “action” loop here so we do not get misled by for example {=}. As the code here is for very much predictable types of input, we hard-code what constitutes math mode opening and closing. At the very beginning, the default key (#1) and the argument as given by the user (#2) are placed right after the \q__cmd_recursion_stop, so that when the recursion ends, the macros __cmd_arg_to_keyvalue_set_default:nn or __cmd_arg_to_keyvalue_set_keyvalue:nn can be used to grab these two items and

set the \ProcessedArgument accordingly.

```

2193 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_auxv:nn #1#2
2194 {
2195     \__cmd_arg_to_keyvalue_loop:w #2
2196     \q__cmd_recursion_tail \q__cmd_recursion_stop {#1} {#2}
2197 }
2198 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_loop:w #1 \q__cmd_recursion_stop
2199 {
2200     \tl_if_head_is_N_type:nTF {#1}
2201     { \__cmd_arg_to_keyvalue_loop_N_type:N }
2202     {
2203         \tl_if_head_is_group:nTF {#1}
2204         { \__cmd_arg_to_keyvalue_loop_group:n }
2205         { \__cmd_arg_to_keyvalue_loop_space:w }
2206     }
2207     #1 \q__cmd_recursion_stop
2208 }
2209 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_loop_group:n #1
2210 { \__cmd_arg_to_keyvalue_loop:w }
2211 \use:n { \cs_new_protected:Npn \__cmd_arg_to_keyvalue_loop_space:w } ~
2212 { \__cmd_arg_to_keyvalue_loop:w }
2213 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_loop_N_type:N #1
2214 {
2215     \__cmd_if_recursion_tail_stop_do:Nn #1
2216     { \__cmd_arg_to_keyvalue_set_default:nn }
2217     \str_if_eq:nnTF {#1} { = }
2218     {
2219         \__cmd_use_i_delimit_by_q_recursion_stop:nw
2220         { \__cmd_arg_to_keyvalue_set_keyvalue:nn }
2221     }
2222     {
2223         \bool_lazy_or:nnTF
2224         { \token_if_math_toggle_p:N #1 }
2225         { \str_if_eq_p:nn {#1} { \{ } }
2226         { \__cmd_arg_to_keyvalue_math:w }
2227         { \__cmd_arg_to_keyvalue_loop:w }
2228     }
2229 }
2230 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_math:w #1 \q__cmd_recursion_stop
2231 {
2232     \tl_if_head_is_N_type:nTF {#1}
2233     { \__cmd_arg_to_keyvalue_math_N_type:N }
2234     {
2235         \tl_if_head_is_group:nTF {#1}
2236         { \__cmd_arg_to_keyvalue_math_group:n }
2237         { \__cmd_arg_to_keyvalue_math_space:w }
2238     }
2239     #1 \q__cmd_recursion_stop
2240 }
2241 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_math_N_type:N #1
2242 {
2243     \__cmd_if_recursion_tail_stop_do:Nn #1
2244     { \__cmd_arg_to_keyvalue_set_default:nn }
2245     \bool_lazy_or:nnTF

```

```

2246      { \token_if_math_toggle_p:N #1 }
2247      { \str_if_eq_p:nn {#1} { \) } }
2248      { \__cmd_arg_to_keyvalue_loop:w }
2249      { \__cmd_arg_to_keyvalue_math:w }
2250  }
2251 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_math_group:n #1
2252  { \__cmd_arg_to_keyvalue_math:w }
2253 \use:n { \cs_new_protected:Npn \__cmd_arg_to_keyvalue_math_space:w } ~
2254  { \__cmd_arg_to_keyvalue_math:w }
2255 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_set_default:nn #1#2
2256  { \tl_set:Nn \ProcessedArgument { #1 = {#2} } }
2257 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_set_keyvalue:nn #1#2
2258  { \tl_set:Nn \ProcessedArgument {#2} }

A utility to allow us to grab the first N-type token without risking brace stripping the
rest of the input.

2259 \cs_new:Npn \__cmd_split_N_head_apply:Nn #1#2
2260  { \exp:w \if_false: { \fi: \__cmd_split_N_head_apply_aux:NNw #1#2 } }
2261 \cs_new:Npn \__cmd_split_N_head_apply_aux:NNw #1#2
2262  {
2263    \exp_after:wN \exp_end:
2264    \exp_after:wN #1 \exp_after:wN #2 \exp_after:wN { \if_false: } \fi:
2265  }
2266
```

(End of definition for `__cmd_arg_to_keyvalue:nn` and others.)

1.12 Access to the argument specification

`__cmd_get_arg_spec_error:N` Provide an informative error when trying to get the argument specification of a non-xparse command or environment.

```

\__cmd_get_arg_spec_error:n
\__cmd_get_arg_spec_error_aux:n
2267 \cs_new_protected:Npn \__cmd_get_arg_spec_error:N #1
2268  {
2269    \bool_set_false:N \l__cmd_environment_bool
2270    \tl_set:Nn \l__cmd_fn_tl {#1}
2271    \__cmd_get_arg_spec_error_aux:n { \cs_if_exist:NTF #1 }
2272  }
2273 \cs_new_protected:Npn \__cmd_get_arg_spec_error:n #1
2274  {
2275    \bool_set_true:N \l__cmd_environment_bool
2276    \str_set:Nx \l__cmd_environment_str {#1}
2277    \__cmd_get_arg_spec_error_aux:n
2278      { \cs_if_exist:cTF { \l__cmd_environment_str } }
2279  }
2280 \cs_new_protected:Npn \__cmd_get_arg_spec_error_aux:n #1
2281  {
2282    #1
2283    {
2284      \msg_error:nnx { cmd } { non-xparse }
2285      { \__cmd_environment_or_command: }
2286    }
2287    {
2288      \msg_error:nnx { cmd } { unknown }
2289      { \__cmd_environment_or_command: }
```

```

2290         }
2291     }

(End of 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.

```

2292 \cs_new_protected:Npn \__cmd_get_arg_spec:NTF #1#2#3
2293 {
2294     \__kernel_cmd_if_xparse:NTF #1
2295     {
2296         \tl_set:Nx \ArgumentSpecification { \tl_item:Nn #1 { 2 } }
2297         #2
2298     }
2299     {[#3]}
2300 }

```

(End of 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.

```

2301 <latexrelease>\cs_new_eq:NN \__cmd_chk_if_free:cs:N \__kernel_chk_if_free:cs:N
2302 <latexrelease>\cs_gset_eq:NN \__kernel_chk_if_free:cs:N \use_none:n

```

\ArgumentSpecification 2303 \tl_new:N \ArgumentSpecification

__cmd_get_arg_spec:N Recovering the argument specification is now trivial.

```

2304 \cs_new_protected:Npn \__cmd_get_arg_spec:N #1
2305 {
2306     \__cmd_get_arg_spec:NTF #1 { }
2307     { \__cmd_get_arg_spec_error:N #1 }
2308 }
2309 \cs_new_protected:Npn \__cmd_get_arg_spec:n #1
2310 {
2311     \exp_args:Nc \__cmd_get_arg_spec:NTF
2312     { environment~ \tl_to_str:n {[#1]} }
2313     { }
2314     { \__cmd_get_arg_spec_error:n {[#1]} }
2315 }

```

(End of 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.

```

2316 \cs_new_protected:Npn \__cmd_show_arg_spec:N #1
2317 {
2318     \__cmd_get_arg_spec:NTF #1
2319     { \tl_show:N \ArgumentSpecification }
2320     { \__cmd_get_arg_spec_error:N #1 }
2321 }

```

```

2322 \cs_new_protected:Npn \__cmd_show_arg_spec:n #1
2323 {
2324   \exp_args:Nc \__cmd_get_arg_spec:NTF
2325   { environment~\tl_to_str:n {#1} }
2326   { \tl_show:N \ArgumentSpecification }
2327   { \__cmd_get_arg_spec_error:n {#1} }
2328 }
```

(End of definition for `__cmd_show_arg_spec:N` and `__cmd_show_arg_spec:n`.)

1.13 Utilities

```
\__cmd_check_definable:nNT
\__cmd_check_definable_aux:nN
```

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.

```

2329 \cs_new_protected:Npn \__cmd_check_definable:nNT #1
2330 { \tl_trim_spaces_apply:nN {#1} \__cmd_check_definable_aux:nN }
2331 \group_begin:
2332   \char_set_catcode_active:n { '?
2333   \cs_new_protected:Npn \__cmd_check_definable_aux:nN #1#2
2334   {
2335     \group_begin:
2336     \tl_if_single_token:nTF {#1}
2337     {
2338       \int_set:Nn \tex_escapechar:D { 92 }
2339       \exp_args:Nx \tl_if_empty:nTF
2340       { \exp_args:No \str_tail:n { \token_to_str:N #1 } }
2341       {
2342         \exp_args:Nx \char_set_lccode:nn
2343         { ' \str_head:n {#1} } { '? }
2344         \tex_lowercase:D { \tl_if_eq:nnTF {#1} } { ? }
2345         { \group_end: \use_iii:nnn }
2346         { \group_end: \use_i:nnn }
2347       }
2348       { \group_end: \use_iii:nnn }
2349     }
2350     { \group_end: \use_ii:nnn }
2351   {
2352     \msg_error:nnxx { cmd } { not-definable }
2353     { \tl_to_str:n {#1} } { \token_to_str:N #2 }
2354   }
2355   {
2356     \msg_error:nnxx { cmd } { not-one-token }
2357     { \tl_to_str:n {#1} } { \token_to_str:N #2 }
2358 }
```

```

2359         }
2360     \group_end:

```

(End of 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.*, `\anything`). `\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.

```

2361 \cs_new_protected:Npn \__cmd_token_if_cs:NTF #1
2362 {
2363     \group_begin:
2364     \int_set:Nn \tex_escapechar:D { 92 }
2365     \exp_args:Nx \tl_if_empty:nTF
2366     { \exp_args:No \str_tail:n { \token_to_str:N #1 } }
2367     { \group_end: \use_i:nn }
2368     { \group_end: \use_i:nn }
2369 }

```

(End of definition for `__cmd_token_if_cs:NTF`.)

Analogue of `\seq_mapthread_function:NNN` for token lists.

```

2370 \cs_new:Npn \__cmd_tl_mapthread_function:NNN #1#2#3
2371 {
2372     \exp_after:wN \exp_after:wN
2373     \exp_after:wN \__cmd_tl_mapthread_loop:w
2374     \exp_after:wN \exp_after:wN
2375     \exp_after:wN #3
2376     \exp_after:wN #1
2377     \exp_after:wN \q_recursion_tail
2378     \exp_after:wN \q_mark
2379     #2
2380     \q_recursion_tail
2381     \q_recursion_stop
2382 }
2383 \cs_new:Npn \__cmd_tl_mapthread_function:nnN #1#2#3
2384 {
2385     \__cmd_tl_mapthread_loop:w #3
2386     #1 \q_recursion_tail \q_mark
2387     #2 \q_recursion_tail \q_recursion_stop
2388 }
2389 \cs_new:Npn \__cmd_tl_mapthread_loop:w #1#2#3 \q_mark #4
2390 {
2391     \quark_if_recursion_tail_stop:n {#2}
2392     \quark_if_recursion_tail_stop:n {#4}
2393     #1 {#2} {#4}
2394     \__cmd_tl_mapthread_loop:w #1#3 \q_mark
2395 }

```

(End of definition for `__cmd_tl_mapthread_function:NNN`, `__cmd_tl_mapthread_function:nnN`, and `__cmd_tl_mapthread_loop:w`.)

__kernel_cmd_if_xparse:NTF 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:nNNnn` (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.

```

2396 \cs_new_protected:Npn \_\_cmd_type_cases:NnnnnF #1 #2 #3 #4 #5 #6
2397 {
2398   \exp_args:Ne \str_case_e:nnF
2399   {
2400     \exp_args:Nf \tl_if_empty:nT { \cs_argument_spec:N #1 }
2401     { \exp_not:N \exp_not:n { \exp_not:e { \tl_head:N #1 } } }
2402   }
2403   {
2404     { \exp_not:N \_\_cmd_start:nNNnnn } {#2}
2405     { \exp_not:N \_\_cmd_start_expandable:nNNNNn } {#3}
2406     { \exp_not:N \_\_cmd_start_env:nnnnn } {#4}
2407     {
2408       \exp_after:wN \exp_not:N
2409       \cs:w environment~
2410       \exp_last_unbraced:Ne \use_none:nnn
2411       { \cs_to_str:N #1 } ~end-aux \cs_end:
2412     } {#5}
2413   }
2414   {#6}
2415 }
2416 \cs_new_protected:Npn \_\_kernel_cmd_if_xparse:NTF #1
2417 {
2418   \_\_cmd_type_cases:NnnnnF #1
2419   { } { } { } { } { \use_iii:nnn }
2420   \use_i:nn
2421 }
```

(End of definition for `__kernel_cmd_if_xparse:NTF`, `__cmd_type_cases:Nnnnn`, and `__cmd_if_xparse_aux:N`.)

__cmd_peek_nonspace:NTF Collect spaces in a loop, and put the collected spaces back in the false branch of a call to `\peek_meaning:NTF` or `\peek_meaning_remove:NTF`.

```

2422 \cs_new_protected:Npn \_\_cmd_peek_nonspace:NTF
2423   { \_\_cmd_peek_nonspace_aux:nNNTF { } \_\_cmd_peek_meaning:NTF }
2424 \cs_new_protected:Npn \_\_cmd_peek_nonspace_remove:NTF
2425   { \_\_cmd_peek_nonspace_aux:nNNTF { } \_\_cmd_peek_meaning_remove:NTF }
2426 \cs_new_protected:Npn \_\_cmd_peek_nonspace_aux:nNNTF #1#2#3#4#5
2427   {
2428     \peek_meaning_remove:NTF \c_space_token
2429     { \_\_cmd_peek_nonspace_aux:nNNTF { #1 ~ } #2 #3 {#4} {#5} }
2430     { #2 #3 {#4} { #5 #1 } }
2431   }
```

(End of definition for `__cmd_peek_nonspace:NTF`, `__cmd_peek_nonspace_remove:NTF`, and `__cmd_peek_nonspace_aux:nNNTF`.)

```
\_\_cmd_peek_meaning:NTF
  \_\_cmd_peek_meaning_remove:NTF
    \_\_cmd_peek_cs_check_equal:NNN
\_\_cmd_peek_meaning_aux:NNTF
\_\_cmd_peek_true_remove:Nw
```

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.

```
2432 \cs_new_protected:Npn \_\_cmd_peek_meaning:NTF
2433   { \_\_cmd_peek_meaning_aux:NNTF \c_false_bool }
2434 \cs_new_protected:Npn \_\_cmd_peek_meaning_remove:NTF
2435   { \_\_cmd_peek_meaning_aux:NNTF \c_true_bool }
2436 \cs_new_protected:Npn \_\_cmd_peek_meaning_aux:NNTF #1#2#3#4
2437   {
2438     \tl_set:Nn \l_\_cmd_tma_t1 {\#3}
2439     \tl_set:Nn \l_\_cmd_tmb_t1 {\#4}
2440     \peek_meaning:NTF #2
2441     {
2442       \token_if_eq_meaning:NNTF #2 \c_group_begin_token
2443         { \_\_cmd_peek_true_remove:Nw #1 }
2444         {
2445           \_\_cmd_token_if_cs:NTF #2
2446             { \_\_cmd_peek_cs_check_equal:NNN #1 #2 }
2447             { \_\_cmd_peek_true_remove:Nw #1 }
2448           }
2449         }
2450         { \l_\_cmd_tmb_t1 }
2451     }
2452 \cs_new_protected:Npn \_\_cmd_peek_cs_check_equal:NNN #1#2#3
2453   {
2454     \str_if_eq:nnTF {\#2} {\#3}
2455       { \_\_cmd_peek_true_remove:Nw #1 }
2456       { \l_\_cmd_tmb_t1 }
2457     #3
2458   }
2459 \cs_new_protected:Npn \_\_cmd_peek_true_remove:Nw #1
2460   {
2461     \bool_if:NTF #1
2462     {
2463       \tex_afterassignment:D \l_\_cmd_tma_t1
2464       \cs_set_eq:NN \_\_cmd_tmp:w
2465     }
2466     { \l_\_cmd_tma_t1 }
2467   }
```

(End of definition for $__cmd_peek_meaning:NTF$ and others.)

1.14 Messages

```
\c_\_cmd_ignore_def_t1
2468 \tl_const:Nn \c_\_cmd_ignore_def_t1
2469   { \\ \\ LaTeX-will-ignore-this-entire-definition. }
```

$__cmd_environment_or_command$:

```
2470 \cs_new:Npn \_\_cmd_environment_or_command:
```

```

2471   {
2472     \bool_if:NTF \l__cmd_environment_bool
2473     { environment ~ ' \l__cmd_environment_str ' }
2474     {
2475       command ~ ,
2476       \exp_args:Nf \tl_trim_spaces:n
2477       { \exp_after:wN \token_to_str:N \l__cmd_fn_tl }
2478     ,
2479   }
2480 }

(End of definition for \__cmd_environment_or_command:.)

Some messages intended as errors when defining commands/environments.

2481 \msg_new:nnn { cmd } { arg-after-body }
2482   { Argument-type~'b'~must~be~last~in~#1. }
2483   {
2484     The~'b'~argument~type~must~come~last~but~it~is~followed~
2485     by~'#2'~in~the~argument~specification.~This~is~not~allowed.
2486     \c__cmd_ignore_def_tl
2487   }
2488 \msg_new:nnn { cmd } { bad-arg-spec }
2489   { Bad~argument~specification~'#2'~for~#1. }
2490   {
2491     The~argument~specification~provided~is~not~valid:~
2492     one~or~more~mandatory~parts~are~missing.
2493     \c__cmd_ignore_def_tl
2494   }
2495 \msg_new:nnn { cmd } { already-defined }
2496   { Command~'#1'~already~defined. }
2497   {
2498     You~have~used~#2~
2499     with~a~command~that~already~has~a~definition. \\ \\
2500     The~existing~definition~of~'#1'~will~not~be~altered.
2501   }
2502 \msg_new:nnn { cmd } { undefined }
2503   { Command ~'#1'~undefined. }
2504   {
2505     You~have~used~#2~
2506     with~a~command~that~was~never~defined.
2507     \c__cmd_ignore_def_tl
2508   }
2509 \msg_new:nnn { cmd } { env-already-defined }
2510   { Environment~'#1'~already~defined. }
2511   {
2512     You~have~used~\NewDocumentEnvironment
2513     with~an~environment~that~already~has~a~definition. \\ \\
2514     The~existing~definition~of~'#1'~will~not~be~altered.
2515   }
2516 \msg_new:nnn { cmd } { env-end-already-defined }
2517   { End~of~environment~'#1'~already~defined. }
2518   {
2519     You~have~used~\NewDocumentEnvironment
2520     with~an~environment~that~already~has~a~definition~for~'end#1'. \\ \\
2521     The~existing~definition~of~'#1'~will~not~be~altered.

```

```

2522 }
2523 \msg_new:nnnn { cmd } { env-undefined }
2524 { Environment~'#1'~undefined. }
2525 {
2526   You~have~used~\RenewDocumentEnvironment
2527   with~an~environment~that~was~never~defined.
2528   \c__cmd_ignore_def_tl
2529 }
2530 \msg_new:nnnn { cmd } { expandable-ending-optional }
2531 { Bad~argument~specification~'#2'~for~#1. }
2532 {
2533   Expandable~commands~must~have~a~final~mandatory~argument~
2534   (or~no~arguments~at~all).~You~cannot~have~a~terminal~optional~
2535   argument~with~expandable~commands.
2536 }
2537 \msg_new:nnnn { cmd } { long-short-mix }
2538 { Invalid~argument~prefix~'+'~in~command~'#1'. }
2539 {
2540   The~arguments~for~an~expandable~command~must~not~involve~short~
2541   arguments~after~long~arguments.~You~have~tried~to~mix~the~two~types~
2542   when~defining~'#1'.
2543 }
2544 \msg_new:nnnn { cmd } { invalid-command-arg }
2545 { Invalid~argument~type~'#2'~in~#1. }
2546 {
2547   The~letter~'#2'~can~only~be~used~in~environment~argument~
2548   specifications,~but~not~for~commands.
2549 \\
2550   LaTeX~will~ignore~the~entire~definition.
2551 }
2552 \msg_new:nnnn { cmd } { invalid-expandable-arg }
2553 { Invalid~argument~type~'#2'~in~#1. }
2554 {
2555   The~letter~'#2'~specifies~an~argument~type~which~cannot~be~used~
2556   in~an~expandable~command.
2557   \c__cmd_ignore_def_tl
2558 }
2559 \msg_new:nnnn { cmd } { invalid-after-optional-expandably }
2560 { Argument~'#2'~invalid~after~optional~arg~in~#1. }
2561 {
2562   The~letter~'#2'~specifies~an~argument~type~which~cannot~be~used~
2563   in~an~expandable~command~after~an~optional~argument.
2564   \c__cmd_ignore_def_tl
2565 }
2566 \msg_new:nnnn { cmd } { invalid-bang }
2567 { Invalid~argument~prefix~'!'~in~#1. }
2568 {
2569   The~prefix~'!'~is~only~allowed~for~trailing~optional~arguments.~
2570   You~tried~to~apply~it~to~'#2'.
2571   \c__cmd_ignore_def_tl
2572 }
2573 \msg_new:nnnn { cmd } { not-definable }
2574 { First~argument~of~'#2'~must~be~a~command. }
2575 {

```

```

2576 The~first~argument~of~'#2'~should~be~the~document~command~that~will~
2577 be~defined.~The~provided~argument~'#1'~is~a~character.~Perhaps~a~
2578 backslash~is~missing?
2579 \c__cmd_ignore_def_tl
2580 }
2581 \msg_new:nnnn { cmd } { not-one-token }
2582 { First~argument~of~'#2'~must~be~a~command. }
2583 {
2584 The~first~argument~of~'#2'~should~be~the~document~command~that~will~
2585 be~defined.~The~provided~argument~'#1'~contains~more~than~one~
2586 token.~Perhaps~a~backslash~is~missing?
2587 \c__cmd_ignore_def_tl
2588 }
2589 \msg_new:nnnn { cmd } { not-single-token }
2590 { Argument~delimiter~'#2'~invalid~in~#1. }
2591 {
2592 The~argument~specification~contains~
2593 \tl_if_empty:nTF{#2}{nothing}{ '#2' }~
2594 in~a~place~
2595 where~a~single~token~is~required.
2596 \c__cmd_ignore_def_tl
2597 }
2598 \msg_new:nnnn { cmd } { forbidden-group-token }
2599 { Argument~delimiter~'#2'~invalid~in~#1. }
2600 {
2601 The~argument~specification~contains~the~implicit~
2602 #3-group~token~'#2'~which~is~not~allowed~as~an~argument~delimiter.
2603 \c__cmd_ignore_def_tl
2604 }
2605 \msg_new:nnnn { cmd } { processor-in-expandable }
2606 { Invalid~argument~prefix~'>'~in~command~'#1'. }
2607 {
2608 The~argument~specification~for~'#1'~contains~the~processor~function~'>{#2}'~.~.
2609 This~is~only~supported~for~robust~commands,~but~not~for~expandable~ones.
2610 \c__cmd_ignore_def_tl
2611 }
2612 \msg_new:nnnn { cmd } { keyval-in-expandable }
2613 { Invalid~argument~prefix~'='~in~command~'#1'. }
2614 {
2615 The~argument~specification~for~'#1'~contains~a~key--value~marker~'={#2}'~.~.
2616 This~is~only~supported~for~robust~commands,~but~not~for~expandable~ones.
2617 \c__cmd_ignore_def_tl
2618 }
2619 \msg_new:nnnn { cmd } { too-many-args }
2620 { Too~many~arguments~for~#1. }
2621 {
2622 The~argument~specification~'#2'~asks~for~more~than~9~arguments.~.
2623 This~cannot~be~implemented.
2624 \c__cmd_ignore_def_tl
2625 }
2626 \msg_new:nnnn { cmd } { two-markers }
2627 { Invalid~argument~prefix~'#2'~in~#1. }
2628 {
2629 The~argument~specification~provided~for~#1~has~two~'#2'~markers~applied~

```

```

2630      to-the-same-argument; one-is-redundant.
2631  }
2632 \msg_new:nnnn { cmd } { unknown-argument-type } % should be unkown-arg-type but dep in xpars...
2633 { Invalid-argument-type-'#2'-in-#1. }
2634 {
2635   The-letter-'#2'-does-not-specify-a-known-argument-type.
2636   \c__cmd_ignore_def_tl
2637 }
2638 \msg_new:nnnn { cmd } { xparse-arg-type }
2639 { Invalid-argument-type-'#2'-in-#1-(requires-xparse). }
2640 {
2641   The-letter-'#2'-specifies-a-known-but-deprecated-argument-type.-
2642   If-you-really-need-it-you-have-to-load-the-xparse-package.
2643   \c__cmd_ignore_def_tl
2644 }

```

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.

```

2645 \msg_new:nnn { cmd } { if-boolean }
2646 { Invalid-argument-{-#1}-to-\iow_char:N\\IfBoolean... }
2647 \msg_new:nnnn { cmd } { default-loop }
2648 { Circular-dependency-in-defaults-of-#1. }
2649 {
2650   The-default-values-of-two-or-more-arguments-of-the-#1-
2651   depend-on-each-other-in-a-way-that-cannot-be-resolved.
2652 }
2653 \msg_new:nnnn { cmd } { missing-required }
2654 { Required-argument-missing-for-#1. }
2655 {
2656   The-#1-expects-one-of-its-arguments-to-start-with-'#2'.-
2657   LaTeX-did-not-find-this-argument-and-will-insert-a-default-value-
2658   for-further-processing.
2659 }
2660 \msg_new:nnnn { cmd } { non-xparse }
2661 { \str_uppercase:n #1-not-defined-using-xparse. }
2662 {
2663   You-have-asked-for-the-argument-specification-for-the-#1,-
2664   but-this-was-not-defined-using-xparse.
2665 }
2666 \msg_new:nnnn { cmd } { arg-split }
2667 { Too-many-'#1'-separators-in-argument. }
2668 {
2669   LaTeX-was-asked-to-split-the-input-'#3'-
2670   at-each-occurrence-of-the-separator-'#1'-into-#2-parts.-
2671   Too-many-separators-were-found.
2672 }
2673 \msg_new:nnnn { cmd } { unknown }
2674 { Unknown-document-#1. }
2675 {
2676   You-have-asked-for-the-argument-specification-for-the-#1,-
2677   but-it-is-not-defined.
2678 }
2679 \msg_new:nnnn { cmd } { verbatim-nl }

```

```

2680 { Verbatim-like~#1-ended~by~end~of~line. }
2681 {
2682   The~verbatim~argument~of~the~#1~cannot~contain~more~than~one~line,~
2683   but~the~end~
2684   of~the~current~line~has~been~reached.~You~may~have~forgotten~the~
2685   closing~delimiter.
2686   \\ \\
2687   LaTeX~will~ignore~'#2'~and~you~may~get~some~additional~
2688   (low-level)~errors.
2689 }
2690 \msg_new:nnn { cmd } { verbatim-tokenized }
2691 { Verbatim-like~#1-illegal~in~argument. }
2692 {
2693   The~#1-takes~a~verbatim~argument~and~should~therefore~normally~
2694   not~be~used~in~arguments~of~other~commands~or~environments.~
2695   LaTeX~found~an~illegal~token~ \tl_if_empty:nF {#3} { (#3)~ }
2696   after~'#2'~and~will~drop~everything~up~to~this~point.
2697   \\ \\
2698   Expect~further~(low-level)~errors.
2699 }

Intended more for information.

2700 \msg_new:nnn { cmd } { define-command }      % should be just ``define'' but dep in xparse
2701 {
2702   Defining~command~#1~
2703   with~sig.~'#2'~\msg_line_context:..
2704 }
2705 \msg_new:nnn { cmd } { define-env }
2706 {
2707   Defining~environment~'#1'~
2708   with~sig.~'#2'~\msg_line_context:..
2709 }
2710 \msg_new:nnn { cmd } { redefine }
2711 {
2712   Redefining~command~#1~
2713   with~sig.~'#2'~\msg_line_context:..
2714 }
2715 \msg_new:nnn { cmd } { redefine-env }
2716 {
2717   Redefining~environment~'#1'~
2718   with~sig.~'#2'~\msg_line_context:..
2719 }
2720 \msg_new:nnn { cmd } { optional-mandatory }
2721 {
2722   Optional~and~mandatory~argument~with~same~delimiter~'#2'.
2723   \\ \\
2724   The~mandatory~argument~specified~with~
2725   '\str_case:nnF{#1}{ {R/r}{r'~or~'R} }{#1}'~has~the~
2726   same~delimiter~'#2'~as~an~earlier~optional~argument.~
2727   It~will~therefore~not~be~possible~to~omit~all~the~earlier~
2728   optional~arguments~when~calling~this~command.
2729   \\ \\
2730   This~may~be~intentional,~but~then~it~might~be~a~mistake.
2731 }
2732 \msg_new:nnn { cmd } { unsupported-let }

```

```

2733 {
2734   The~command~'#1'~was~undefined~but~not~the~associated~commands~
2735   '#1~code'~and/or~'#1~defaults'.~Maybe~you~tried~using~
2736   \iow_char:N\\let.~This~may~lead~to~an~infinite~loop.
2737 }

```

1.15 User functions

The user functions are more or less just the internal functions renamed.

\BooleanFalse Design-space names for the Boolean values.

```

2738 \cs_new_eq:NN \BooleanFalse \c_false_bool
2739 \cs_new_eq:NN \BooleanTrue \c_true_bool

```

(End of definition for \BooleanFalse and \BooleanTrue.)

\NewDocumentCommand \RenewDocumentCommand \ProvideDocumentCommand \DeclareDocumentCommand The user macros are pretty simple wrappers around the internal ones. There is however a check that the first argument is a single token, possibly surrounded by spaces (hence the strange \use:nnn), and is definable.

```

2740 \cs_new_protected:Npn \NewDocumentCommand #1#2#3
2741 {
2742   \__cmd_check_definable:nNT {#1} \NewDocumentCommand
2743   {
2744     \cs_if_exist:NTF #1
2745     {
2746       \msg_error:nnxx { cmd } { already-defined }
2747       { \use:nnn \token_to_str:N #1 { } }
2748       { \token_to_str:N \NewDocumentCommand }
2749     }
2750     { \__cmd_declare_cmd:Nnn #1 {#2} {#3} }
2751   }
2752 }
2753 \cs_new_protected:Npn \RenewDocumentCommand #1#2#3
2754 {
2755   \__cmd_check_definable:nNT {#1} \RenewDocumentCommand
2756   {
2757     \cs_if_exist:NTF #1
2758     {
2759       \__cmd_declare_cmd:Nnn #1 {#2} {#3}
2760       \msg_error:nnxx { cmd } { undefined }
2761       { \use:nnn \token_to_str:N #1 { } }
2762       { \token_to_str:N \RenewDocumentCommand }
2763     }
2764   }
2765 }
2766 \cs_new_protected:Npn \ProvideDocumentCommand #1#2#3
2767 {
2768   \__cmd_check_definable:nNT {#1} \ProvideDocumentCommand
2769   { \cs_if_exist:NF #1 { \__cmd_declare_cmd:Nnn #1 {#2} {#3} } }
2770 }
2771 \cs_new_protected:Npn \DeclareDocumentCommand #1#2#3
2772 {
2773   \__cmd_check_definable:nNT {#1} \DeclareDocumentCommand
2774   { \__cmd_declare_cmd:Nnn #1 {#2} {#3} }

```

2775 }

(End of definition for \NewDocumentCommand and others.)

\NewDocumentEnvironment Very similar for environments.

```
2776 \cs_new_protected:Npn \NewDocumentEnvironment #1#2#3#4
2777 {
2778     \cs_if_exist:cTF {#1}
2779     { \msg_error:nnx { cmd } { env-already-defined } {#1} }
2780     {
2781         \cs_if_exist:cTF { end #1 }
2782         { \msg_error:nnx { cmd } { env-end-already-defined } {#1} }
2783         { \__cmd_declare_env:nnnn {#1} {#2} {#3} {#4} }
2784     }
2785 }
2786 \cs_new_protected:Npn \RenewDocumentEnvironment #1#2#3#4
2787 {
2788     \cs_if_exist:cTF {#1}
2789     { \__cmd_declare_env:nnnn {#1} {#2} {#3} {#4} }
2790     { \msg_error:nnx { cmd } { env-undefined } {#1} }
2791 }
2792 \cs_new_protected:Npn \ProvideDocumentEnvironment #1#2#3#4
2793 { \cs_if_exist:cF {#1} { \__cmd_declare_env:nnnn {#1} {#2} {#3} {#4} } }
2794 \cs_new_protected:Npn \DeclareDocumentEnvironment #1#2#3#4
2795 { \__cmd_declare_env:nnnn {#1} {#2} {#3} {#4} }
```

(End of definition for \NewDocumentEnvironment and others.)

\NewExpandableDocumentCommand The expandable versions are essentially the same as the basic functions. The strange \use:nnn is there in case #1 is surrounded with spaces, as can happen with usual document catcodes in \RenewExpandableDocumentCommand { \! } ...

```
2796 \cs_new_protected:Npn \NewExpandableDocumentCommand #1#2#3
2797 {
2798     \__cmd_check_definable:nNT {#1} \NewExpandableDocumentCommand
2799     {
2800         \cs_if_exist:NTF #1
2801         {
2802             \msg_error:nnxx { cmd } { already-defined }
2803             { \use:nnn \token_to_str:N #1 { } }
2804             { \token_to_str:N \NewExpandableDocumentCommand }
2805         }
2806         { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
2807     }
2808 }
2809 \cs_new_protected:Npn \RenewExpandableDocumentCommand #1#2#3
2810 {
2811     \__cmd_check_definable:nNT {#1} \RenewExpandableDocumentCommand
2812     {
2813         \cs_if_exist:NTF #1
2814         { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
2815         {
2816             \msg_error:nnxx { cmd } { undefined }
2817             { \use:nnn \token_to_str:N #1 { } }
2818             { \token_to_str:N \RenewExpandableDocumentCommand }
```

```

2819     }
2820   }
2821 }
2822 \cs_new_protected:Npn \ProvideExpandableDocumentCommand #1#2#3
2823 {
2824   \__cmd_check_definable:nNT {#1} \ProvideExpandableDocumentCommand
2825   {
2826     \cs_if_exist:NF #1
2827     { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
2828   }
2829 }
2830 \cs_new_protected:Npn \DeclareExpandableDocumentCommand #1#2#3
2831 {
2832   \__cmd_check_definable:nNT {#1} \DeclareExpandableDocumentCommand
2833   { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
2834 }

```

(End of 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.

```

2835 \cs_new:Npn \IfBooleanTF #1
2836 {
2837   \tl_if_single:nF {#1}
2838   { \prg_break:n { \use:n } }
2839   \tl_if_single_token:nF #1
2840   { \prg_break:n { \use:n } }
2841   \token_if_eq_meaning:NNT #1 \c_true_bool
2842   { \prg_break:n { \use_ii:nnn } }
2843   \token_if_eq_meaning:NNT #1 \c_false_bool
2844   { \prg_break:n { \use_iii:nnn } }
2845   \prg_break:n { \use:n }
2846   \prg_break_point:
2847   {
2848     \msg_expandable_error:nnn { cmd } { if-boolean } {#1}
2849     \use_ii:nn
2850   }
2851 }
2852 \cs_new:Npn \IfBooleanT #1#2 { \IfBooleanTF {#1} {#2} { } }
2853 \cs_new:Npn \IfBooleanF #1 { \IfBooleanTF {#1} { } }
```

(End of definition for `\IfBooleanT`, `\IfBooleanF`, and `\IfBooleanTF`.)

`\IfNoValueT` Simple re-naming.
`\IfNoValueF` `\cs_new_eq:NN \IfNoValueF \tl_if_novalue:nF`
`\IfNoValueTF` `\cs_new_eq:NN \IfNoValueT \tl_if_novalue:nT`
`\cs_new_eq:NN \IfNoValueTF \tl_if_novalue:nTF`

(End of definition for \IfNoValueT, \IfNoValueF, and \IfNoValueTF.)

\IfValueT Inverted logic.
2857 \cs_new:Npn \IfValueF { \tl_if_novalue:nT }
2858 \cs_new:Npn \IfValueT { \tl_if_novalue:nF }
2859 \cs_new:Npn \IfValueTF #1#2#3 { \tl_if_novalue:nTF {#1} {#3} {#2} }

(End of definition for \IfValueT, \IfValueF, and \IfValueTF.)

\IfBlankT Another simple re-naming.
2860 ⟨latexrelease⟩\IncludeInRelease{2022/06/01}%
\IfBlankF 2861 ⟨latexrelease⟩ {\IfBlankTF}{Testing-for-empty-or-blank}%
2862 \cs_new_eq:NN \IfBlankF \tl_if_blank:nF
2863 \cs_new_eq:NN \IfBlankT \tl_if_blank:nT
2864 \cs_new_eq:NN \IfBlankTF \tl_if_blank:nTF
2865 ⟨latexrelease⟩\EndIncludeInRelease
2866 ⟨latexrelease⟩\IncludeInRelease{2021/11/15}%
2867 ⟨latexrelease⟩ {\IfBlankTF}{Testing-for-empty-or-blank}%
2868 ⟨latexrelease⟩\cs_undefine:N \IfBlankF
2869 ⟨latexrelease⟩\cs_undefine:N \IfBlankT
2870 ⟨latexrelease⟩\cs_undefine:N \IfBlankTF
2871 ⟨latexrelease⟩
2872 ⟨latexrelease⟩\EndIncludeInRelease

(End of definition for \IfBlankT, \IfBlankF, and \IfBlankTF.)

\ProcessedArgument Processed arguments are returned using this name, which is reserved here although the definition will change.
2873 \tl_new:N \ProcessedArgument

(End of definition for \ProcessedArgument.)

\ReverseBoolean Simple copies.
\SplitArgument 2874 \cs_new_eq:NN \ReverseBoolean __cmd_bool_reverse:N
\SplitList 2875 \cs_new_eq:NN \SplitArgument __cmd_split_argument:nnn
\TrimSpaces 2876 \cs_new_eq:NN \SplitList __cmd_split_list:nn
2877 \cs_new_eq:NN \TrimSpaces __cmd_trim_spaces:n

(End of definition for \ReverseBoolean and others.)

\ProcessList To support \SplitList.
2878 \cs_new_eq:NN \ProcessList \tl_map_function:nN

(End of definition for \ProcessList.)

\GetDocumentCommandArgSpec More simple mappings, with a check that the argument is a single control sequence or active character.
\GetDocumentEnvironmentArgSpec
\ShowDocumentCommandArgSpec 2879 \cs_new_protected:Npn \GetDocumentCommandArgSpec #1
2880 {
 2881 __cmd_check_definable:nNT {#1} \GetDocumentCommandArgSpec
 2882 { __cmd_get_arg_spec:N #1 }
 2883 }
2884 \cs_new_eq:NN \GetDocumentEnvironmentArgSpec __cmd_get_arg_spec:n
2885 \cs_new_protected:Npn \ShowDocumentCommandArgSpec #1
2886 {

```

2887     \__cmd_check_definable:nNT {#1} \ShowDocumentCommandArgSpec
2888     { \__cmd_show_arg_spec:N #1 }
2889   }
2890 \cs_new_eq:NN \ShowDocumentEnvironmentArgSpec \__cmd_show_arg_spec:n

(End of definition for \GetDocumentCommandArgSpec and others.)

```

Finally as promised, restore __kernel_chk_if_free_cs:N:

```

2891 \cs_gset_eq:NN \__kernel_chk_if_free_cs:N \__cmd_chk_if_free_cs:N
2892 \cs_undefine:N \__cmd_chk_if_free_cs:N
2893 \cs_gset_eq:NN \__kernel_chk_if_free_cs:N \__cmd_chk_if_free_cs:N
2894 \IncludeInRelease{0000/00/00}{\ltcmd}%
2895 \cs_undefine:N \__cmd_chk_if_free_cs:N
2896 \cs_gset_eq:NN \__kernel_chk_if_free_cs:N \__cmd_chk_if_free_cs:N
2897 \EndModuleRelease
2898 \ExplSyntaxOff

```

Now in `\textrm{latexrelease}` mode, redefine `\NewDocumentCommand` to not complain on commands already defined.

```

2899 \cs_gset_eq:NN \__kernel_chk_if_free_cs:N \__cmd_chk_if_free_cs:N
2900 \catcode`\^^@=\@texrelease@catcode@null\relax
2901 \relax

```

We need to stop DocStrip treating `\@C@=` in a special way at this point.

```

2902 \cs_gset_eq:NN \__kernel_chk_if_free_cs:N \__cmd_chk_if_free_cs:N
2903 \catcode`\^^@=\@texrelease@catcode@null\relax
2904 \relax

```

File h

lthooks.dtx

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_E 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_E interfaces

2.1.1 Declaring hooks

With a few exceptions, hooks have to be declared before they can be used. The exceptions are the generic hooks for commands and environments (executed at `\begin` and `\end`), and the hooks run when loading files (see section 3.1).

`\NewHook \NewHook {⟨hook⟩}`

Creates a new `⟨hook⟩`. If this hook is declared within a package it is suggested that its name is always structured as follows: `⟨package-name⟩/⟨hook-name⟩`. If necessary you can further subdivide the name by adding more / parts. If a hook name is already taken, an error is raised and the hook is not created.

The `⟨hook⟩` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\NewReversedHook \NewReversedHook {⟨hook⟩}`

Like `\NewHook` declares a new `⟨hook⟩`. the difference is that the code chunks for this hook are in reverse order by default (those added last are executed first). Any rules for the hook are applied after the default ordering. See sections 2.3 and 2.4 for further details.

The `⟨hook⟩` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

```
\NewMirroredHookPair \NewMirroredHookPair {\langle hook-1\rangle} {\langle hook-2\rangle}
```

A shorthand for `\NewHook{\langle hook-1\rangle}\NewReversedHook{\langle hook-2\rangle}`.

The `\langle hook\rangle` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

```
\NewHookWithArguments \NewHookWithArguments {\langle hook\rangle} {\langle number\rangle}
```

Creates a new `\langle hook\rangle` whose code takes `\langle number\rangle` arguments, and otherwise works exactly like `\NewHook`. Section 2.7 explains hooks with arguments.

The `\langle hook\rangle` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

```
\NewReversedHookWithArguments \NewReversedHookWithArguments {\langle hook\rangle} {\langle number\rangle}
```

Like `\NewReversedHook`, but creates a hook whose code takes `\langle number\rangle` arguments. Section 2.7 explains hooks with arguments.

The `\langle hook\rangle` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

```
\NewMirroredHookPairWithArguments \NewMirroredHookPairWithArguments {\langle hook-1\rangle} {\langle hook-2\rangle} {\langle number\rangle}
```

A shorthand for `\NewHookWithArguments{\langle hook-1\rangle}{\langle number\rangle}`

`\NewReversedHookWithArguments{\langle hook-2\rangle}{\langle number\rangle}`. Section 2.7 explains hooks with arguments.

The `\langle hook\rangle` 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 {\langle hook\rangle}
```

After this declaration³ the `\langle hook\rangle` is no longer usable: Any further attempt to add 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 `\langle hook\rangle` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

```
\ActivateGenericHook \ActivateGenericHook {\langle hook\rangle}
```

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.

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.

2.1.3 Using hooks in code

\UseHook \UseHook {\hook}

Execute the code stored in the *hook*.

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.

\UseHookWithArguments \UseHookWithArguments {\hook} {\number} {\arg_1} ... {\arg_n}

Execute the code stored in the *hook* and pass the arguments {\arg_1} through {\arg_n} to the *hook*. Otherwise, it works exactly like \UseHook. The *number* should be the number of arguments declared for the hook. If the hook is not declared, this command does nothing and it will remove *number* items from the input. Section 2.7 explains hooks with arguments.

The *hook* cannot be specified using the dot-syntax. A leading . is treated literally.

\UseOneTimeHook \UseOneTimeHook {\hook}

Some hooks are only used (and can be only used) in one place, for example, those in \begin{document} or \end{document}. From that point onwards, 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.

\UseOneTimeHookWithArguments \UseOneTimeHookWithArguments {\hook} {\number} {\arg_1} ... {\arg_n}

Works exactly like \UseOneTimeHook, but passes arguments {\arg_1} through {\arg_n} to the *hook*. The *number* should be the number of arguments declared for the hook. If the hook is not declared, this command does nothing and it will remove *number* items from the input.

It should be noted that after a one-time hook is used, it is no longer possible to use \AddToHookWithArguments or similar with that hook. \AddToHook continues to work as normal. Section 2.7 explains hooks with arguments.

2.1.4 Updating code for hooks

`\AddToHook \AddToHook {\hook}{[label]}{\code}`

Adds `\code` to the `\hook` labeled by `\label`. When the optional argument `\label` is not provided, the `\default label` is used (see section 2.1.5). If `\AddToHook` is used in a package/class, the `\default label` is the package/class name, otherwise it is `top-level` (the `top-level` label is treated differently: see section 2.1.6).

If there already exists code under the `\label` then the new `\code` is appended to the existing one (even if this is a reversed hook). If you want to replace existing code under the `\label`, first apply `\RemoveFromHook`.

The hook doesn't have to exist for code to be added to it. However, if it is not declared, then obviously the added `\code` will never be executed. This allows for hooks to work regardless of package loading order and enables packages to add to hooks from other packages without worrying whether they are actually used in the current document. See section 2.1.8.

The `\hook` and `\label` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\AddToHookWithArguments \AddToHookWithArguments {\hook}{[label]}{\code}`

Works exactly like `\AddToHook`, except that the `\code` can access the arguments passed to the hook using `#1`, `#2`, ..., `#n` (up to the number of arguments declared for the hook). If the `\code` should contain *parameter tokens* (`#`) that are not supposed to be understood as the arguments of the hook, such tokens should be doubled. For example, with `\AddToHook` one can write:

`\AddToHook{myhook}{\def\foo#1>Hello, #1!}`

but to achieve the same with `\AddToHookWithArguments`, one should write:

`\AddToHookWithArguments{myhook}{\def\foo##1>Hello, ##1!}`

because in the latter case, `#1` refers to the first argument of the hook `myhook`. Section 2.7 explains hooks with arguments.

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 {\langle hook\rangle}{\langle code\rangle}
```

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{\langle hook\rangle}{\langle code-1\rangle}\AddToHookNext{\langle hook\rangle}{\langle code-2\rangle}}
```

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.

⁴There is no mechanism to reorder such code chunks (or delete them).

\AddToHookNextWithArguments \AddToHookNextWithArguments {*hook*}{{*code*}}

Works exactly like **\AddToHookNext**, but the *code* can contain references to the arguments of the *hook* as described for **\AddToHookWithArguments** above. Section 2.7 explains hooks with arguments.

The *hook* 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 label* because then the package or class name is automatically used, which is helpful if rules are needed, and avoids mistyping the *label*.

Using an explicit *label* 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 *label* 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 *hook* and *label* 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 of the *hook* or *label* contains a non-expandable non-character token, a low-level `TEX` error is raised (namely, the *hook* is expanded using `TEX`'s `\csname...``\endcsname`, as such, Unicode characters are allowed in *hook* and *label* 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 *labels* 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 *hook* and in a *label*. If the *hook* name or the *label* consist just of a single dot (.), or starts with a dot followed by a slash (./) then the dot denotes the *default label* (usually the current package or class name—see **\SetDefaultHookLabel**). A “.” or “./” anywhere else in a *hook* or in *label* 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  {mypackage/hook}
\AddToHook {mypackage/hook} [mypackage]{code}
\AddToHook {mypackage/hook} [mypackage/sub]{code}
\DeclareHookRule{begindocument}{mypackage}{before}{babel}
\AddToHook {file/foo.tex/after}{code}           % unchanged
```

The *<default label>* 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 *<default label>*. This may have exceptions—see `\PushDefaultHookLabel`.

This syntax is available in all *<label>* arguments and most *<hook>* 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 *<default label>* takes place when the hook command is executed, so actions that are somehow executed after the package ends will have the wrong *<default label>* 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 the code in logical parts, but still use the main package name as the *<label>*, then the *<default label>* can be set using `\PushDefaultHookLabel{...}`...`\PopDefaultHookLabel` or `\SetDefaultHookLabel{...}`.

```
\PushDefaultHookLabel \PushDefaultHookLabel {\<default label>}
\PopDefaultHookLabel   <code>
\PopDefaultHookLabel
```

\PushDefaultHookLabel sets the current *<default label>* to be used in *<label>* arguments, or when replacing a leading “.” (see above). \PopDefaultHookLabel reverts the *<default label>* to its previous value.

Inside a package or class, the *<default label>* is equal to the package or class name, unless explicitly changed. Everywhere else, the *<default label>* is **top-level** (see section 2.1.6) unless explicitly changed.

The effect of \PushDefaultHookLabel holds until the next \PopDefaultHookLabel. \usepackage (and \RequirePackage and \documentclass) internally use

```
\PushDefaultHookLabel{\<package name>}
  <package code>
\PopDefaultHookLabel
```

to set the *<default label>* for the package or class file. Inside the *<package code>* the *<default label>* can also be changed with \SetDefaultHookLabel. \input and other file input-related commands from the L^AT_EX kernel do not use \PushDefaultHookLabel, so code within files loaded by these commands does *not* get a dedicated *<label>*! (that is, the *<default label>* is the current active one when the file was loaded.)

Packages that provide their own package-like interfaces (TikZ’s \usetikzlibrary, for example) can use \PushDefaultHookLabel and \PopDefaultHookLabel to set dedicated labels and to emulate \usepackage-like hook behavior within those contexts.

The **top-level** label is treated differently, and is reserved to the user document, so it is not allowed to change the *<default label>* to **top-level**.

```
\SetDefaultHookLabel \SetDefaultHookLabel {\<default label>}
```

Similarly to \PushDefaultHookLabel, sets the current *<default label>* to be used in *<label>* arguments, or when replacing a leading “.”. The effect holds until the label is changed again or until the next \PopDefaultHookLabel. The difference between \PushDefaultHookLabel and \SetDefaultHookLabel is that the latter does not save the current *<default label>*.

This command is useful when a large package is composed of several smaller packages, but all should have the same *<label>*, so \SetDefaultHookLabel can be used at the beginning of each package file to set the correct label.

\SetDefaultHookLabel is not allowed in the main document, where the *<default label>* is **top-level** and there is no \PopDefaultHookLabel to end its effect. It is also not allowed to change the *<default label>* to **top-level**.

2.1.6 The top-level label

The **top-level** label, assigned to code added from the main document, is different from other labels. Code added to hooks (usually \AtBeginDocument) in the preamble is almost always to change something defined by a package, so it should go at the very end of the hook.

Therefore, code added in the **top-level** is always executed at the end of the hook, regardless of where it was declared. If the hook is reversed (see \NewReversedHook), the **top-level** chunk is executed at the very beginning instead.

Rules regarding `top-level` have no effect: if a user wants to have a specific set of rules for a code chunk, they should use a different label to said code chunk, and provide a rule for that label instead.

The `top-level` label is exclusive for the user, so trying to add code with that label from a package results in an error.

2.1.7 Defining relations between hook code

The default assumption is that code added to hooks by different packages are independent and the order in which they are executed is irrelevant. While this is true in many cases it is obviously false in others.

Before the hook management system was introduced packages had to take elaborate precaution to determine of some other package got loaded as well (before or after) and find some ways to alter its behavior accordingly. In addition it was often the user's responsibility to load packages in the right order so that code added to hooks got added in the right order and some cases even altering the loading order wouldn't resolve the conflicts.

With the new hook management system it is now possible to define rules (i.e., relationships) between code chunks added by different packages and explicitly describe in which order they should be processed.

```
\DeclareHookRule \DeclareHookRule {\langle hook\rangle}{\langle label1\rangle}{\langle relation\rangle}{\langle label2\rangle}
```

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

Currently, the supported relations are the following:

`before` or `<` Code for $\langle label1 \rangle$ comes before code for $\langle label2 \rangle$.

`after` or `>` Code for $\langle label1 \rangle$ comes after code for $\langle label2 \rangle$.

`incompatible-warning` Only code for either $\langle label1 \rangle$ or $\langle label2 \rangle$ can appear for that hook (a way to say that two packages—or parts of them—are incompatible). A warning is raised if both labels appear in the same hook.

`incompatible-error` Like `incompatible-warning` but instead of a warning a L^AT_EX error is raised, and the code for both labels are dropped from that hook until the conflict is resolved.

`voids` Code for $\langle label1 \rangle$ overwrites code for $\langle label2 \rangle$. More precisely, code for $\langle label2 \rangle$ is dropped for that hook. This can be used, for example if one package is a superset in functionality of another one and therefore wants to undo code in some hook and replace it with its own version.

`unrelated` The order of code for $\langle label1 \rangle$ and $\langle label2 \rangle$ is irrelevant. This rule is there to undo an incorrect rule specified earlier.

There can only be a single relation between two labels for a given hook, i.e., a later `\DeclareHookrule` overwrites any previous declaration.

The $\langle hook \rangle$ and $\langle label \rangle$ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

```
\ClearHookRule \ClearHookRule{\hook}{\label1}{\label2}
```

Syntactic sugar for saying that $\langle\text{label1}\rangle$ and $\langle\text{label2}\rangle$ are unrelated for the given $\langle\text{hook}\rangle$.

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

This sets up a relation between $\langle\text{label1}\rangle$ and $\langle\text{label2}\rangle$ for all hooks unless overwritten by a specific rule for a hook. Useful for cases where one package has a specific relation to some other package, e.g., is `incompatible` or always needs a special ordering `before` or `after`. (Technically it is just a shorthand for using `\DeclareHookRule` with `??` as the hook name.)

Declaring default rules is only supported in the document preamble.⁵

The $\langle\text{label}\rangle$ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

2.1.8 Querying hooks

Simpler data types, like token lists, have three possible states; they can:

- exist and be empty;
- exist and be non-empty; and
- not exist (in which case emptiness doesn't apply);

Hooks are a bit more complicated: a hook may exist or not, and independently it may or may not be empty. This means that even a hook that doesn't exist may be non-empty and it can also be disabled.

This seemingly strange state may happen when, for example, package *A* defines hook *A/foo*, and package *B* adds some code to that hook. However, a document may load package *B* before package *A*, or may not load package *A* at all. In both cases some code is added to hook *A/foo* without that hook being defined yet, thus that hook is said to be non-empty, whereas it doesn't exist. Therefore, querying the existence of a hook doesn't imply its emptiness, neither does the other way around.

Given that code or rules can be added to a hook even if it doesn't physically exist yet, means that a querying its existence has no real use case (in contrast to other variables that can only be update if they have already been declared). For that reason only the test for emptiness has a public interface.

A hook is said to be empty when no code was added to it, either to its permanent code pool, or to its “next” token list. The hook doesn't need to be declared to have code added to its code pool. A hook is said to exist when it was declared with `\NewHook` or some variant thereof. Generic hooks such as `file` and `env` hooks are automatically declared when code is added to them.

```
\IfHookEmptyTF * \IfHookEmptyTF {\hook} {\truecode} {\falsecode}
```

Tests if the $\langle\text{hook}\rangle$ is empty (*i.e.*, no code was added to it using either `\AddToHook` or `\AddToHookNext`) or such code was removed again (via `\RemoveFromHook`), and branches to either $\langle\text{true code}\rangle$ or $\langle\text{false code}\rangle$ depending on the result.

The $\langle\text{hook}\rangle$ cannot be specified using the dot-syntax. A leading `.` is treated literally.

⁵Trying to do so, e.g., via `\DeclareHookRule` with `??` has bad side-effects and is not supported (though not explicitly caught for performance reasons).

2.1.9 Displaying hook code

If one has to adjust the code execution in a hook using a hook rule it is helpful to get some information about the code associated with a hook, its current order and the existing rules.

```
\ShowHook \ShowHook {{hook}}
\LogHook \LogHook {{hook}}
```

Displays information about the *hook* such as

- the code chunks (and their labels) added to it,
- any rules set up to order them,
- the computed order in which the chunks are executed,
- any code executed on the next invocation only.

\LogHook prints the information to the .log file, and \ShowHook prints them to the terminal/command window and starts TeX's prompt (only in \errorstopmode) to wait for user action.

The *hook* can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

Suppose a hook `example-hook` whose output of \ShowHook{example-hook} is:

```
1   -> The hook 'example-hook':
2   > Code chunks:
3   >     foo -> [code from package 'foo']
4   >     bar -> [from package 'bar']
5   >     baz -> [package 'baz' is here]
6   > Document-level (top-level) code (executed last):
7   >     -> [code from 'top-level']
8   > Extra code for next invocation:
9   >     -> [one-time code]
10  > Rules:
11  >     foo|baz with relation >
12  >     baz|bar with default relation <
13  > Execution order (after applying rules):
14  >     baz, foo, bar.
```

In the listing above, lines 3 to 5 show the three code chunks added to the hook and their respective labels in the format

(label) -> *(code)*

Line 7 shows the code chunk added by the user in the main document (labeled `top-level`) in the format

```
Document-level (top-level) code (executed {first/last}):
-> {top-level code}
```

This code will be either the first or last code executed by the hook (`last` if the hook is normal, `first` if it is reversed). This chunk is not affected by rules and does not take part in sorting.

Line 9 shows the code chunk for the next execution of the hook in the format

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

This code will be used and disappear at the next `\UseHook{example-hook}`, in contrast to the chunks mentioned earlier, which can only be removed from that hook by doing `\RemoveFromHook{\label}{example-hook}`.

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

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

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` `\DebugHooksOn`

`\DebugHooksOff`

Turn the debugging of hook code on or off. This displays most changes made to the hook data structures. The output is rather coarse and not really intended for normal use.

2.2 L3 programming layer (`expl3`) interfaces

This is a quick summary of the L^AT_EX3 programming interfaces for use with packages written in `expl3`. In contrast to the L^AT_EX 2_E interfaces they always use mandatory arguments only, e.g., you always have to specify the $\langle label \rangle$ for a code chunk. We therefore suggest to use the declarations discussed in the previous section even in `expl3` packages, but the choice is yours.

`\hook_new:n` `\hook_new:n {\langle hook \rangle}`
`\hook_new_reversed:n` `\hook_new_reversed:n {\langle hook \rangle}`
`\hook_new_pair:nn` `\hook_new_pair:nn {\langle hook-1 \rangle} {\langle hook-2 \rangle}`

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_new_with_args:nn` `\hook_new_with_args:nn {\langle hook \rangle} {\langle number \rangle}`
`\hook_new_reversed_with_args:nn` `\hook_new_reversed_with_args:nn {\langle hook \rangle} {\langle number \rangle}`
`\hook_new_pair_with_args:nnn` `\hook_new_pair_with_args:nnn {\langle hook-1 \rangle} {\langle hook-2 \rangle} {\langle number \rangle}`

Creates a new $\langle hook \rangle$ with normal or reverse ordering of code chunks, that takes $\langle number \rangle$ arguments from the input stream when it is used. `\hook_new_pair_with_args:nn` creates a pair of such hooks with $\{\langle hook-2 \rangle\}$ being a reversed hook. If a hook name is already taken, an error is raised and the hook is not created.

The $\langle hook \rangle$ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

```
\hook_disable_generic:n \hook_disable_generic:n {<hook>}
```

Marks {<hook>} 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 [lcmdhooks-doc](#)) if they receive code.

The <hook> can be specified using the dot-syntax to denote the current package name. See section [2.1.5](#).

```
\hook_activate_generic:n \hook_activate_generic:n {<hook>}
```

This is like \hook_new:n but it does nothing if the hook was previously declared with \hook_new:n. This declaration should be used only in special situations, e.g., when a command from another package needs to be altered and it is not clear whether a generic cmd hook (for that command) has been previously explicitly declared.

Normally \hook_new:n should be used instead of this.

```
\hook_use:n \hook_use:n {<hook>}  
\hook_use:nnw \hook_use:nnw {<hook>} {<number>} {<arg1>} ... {<argn>}
```

Executes the {<hook>} code followed (if set up) by the code for next invocation only, then empties that next invocation code. \hook_use:nnw should be used for hooks declared with arguments, and should be followed by as many brace groups as the declared number of arguments. The <number> should be the number of arguments declared for the hook. If the hook is not declared, this command does nothing and it will remove <number> items from the input.

The <hook> cannot be specified using the dot-syntax. A leading . is treated literally.

```
\hook_use_once:n \hook_use_once:n {<hook>}  
\hook_use_once:nnw \hook_use_once:nnw {<hook>} {<number>} {<arg1>} ... {<argn>}
```

Changes the {<hook>} status so that from now on any addition to the hook code is executed immediately. Then execute any {<hook>} code already set up. \hook_use_once:nnw should be used for hooks declared with arguments, and should be followed by as many brace groups as the declared number of arguments. The <number> should be the number of arguments declared for the hook. If the hook is not declared, this command does nothing and it will remove <number> items from the input.

The <hook> cannot be specified using the dot-syntax. A leading . is treated literally.

```
\hook_gput_code:nnn \hook_gput_code:nnn {<hook>} {<label>} {<code>}  
\hook_gput_code_with_args:nnn \hook_gput_code_with_args:nnn {<hook>} {<label>} {<code>}
```

Adds a chunk of <code> to the <hook> labeled <label>. If the label already exists the <code> is appended to the already existing code.

If \hook_gput_code_with_args:nnn is used, the <code> can access the arguments passed to \hook_use:nnw (or \hook_use_once:nnw) with #1, #2, ..., #n (up to the number of arguments declared for the hook). In that case, if an actual parameter token should be added to the code, it should be doubled.

If code is added to an external <hook> (of the kernel or another package) then the convention is to use the package name as the <label> not some internal module name or some other arbitrary string.

The <hook> and <label> can be specified using the dot-syntax to denote the current package name. See section [2.1.5](#).

```
\hook_gput_next_code:nn          \hook_gput_next_code:nn {\langle hook\rangle} {\langle code\rangle}
\hook_gput_next_code_with_args:nn
```

Adds a chunk of *code* for use only in the next invocation of the *hook*. Once used it is gone.

If `\hook_gput_next_code_with_args:nn` is used, the *code* can access the arguments passed to `\hook_use:nnw` (or `\hook_use_once:nnw`) with #1, #2, ..., #n (up to the number of arguments declared for the hook). In that case, if an actual parameter token should be added to the code, it should be doubled.

This is simpler than `\hook_gput_code:nnn`, the code is simply appended to the hook in the order of declaration at the very end, i.e., after all standard code for the hook got executed. Thus if one needs to undo what the standard does one has to do that as part of *code*.

The *hook* can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

```
\hook_gclear_next_code:n \hook_gclear_next_code:n {\langle hook\rangle}
Undo any earlier \hook_gput_next_code:nn.
```

```
\hook_gremove_code:nn \hook_gremove_code:nn {\langle hook\rangle} {\langle label\rangle}
```

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 {\langle hook\rangle} {\langle label1\rangle} {\langle relation\rangle} {\langle label2\rangle}
```

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 {\langle hook\rangle} {\langle true code\rangle} {\langle false code\rangle}
```

`\hook_if_empty:nTF *` Tests if the *hook* is empty (*i.e.*, no code was added to it using either `\AddToHook` or `\AddToHookNext`), and branches to either *true code* or *false code* depending on the result.

The *hook* *cannot* be specified using the dot-syntax. A leading . is treated literally.

```
\hook_show:n \hook_show:n {\hook}
\hook_log:n \hook_log:n {\hook}
```

Displays information about the *hook* such as

- the code chunks (and their labels) added to it,
- any rules set up to order them,
- the computed order in which the chunks are executed,
- any code executed on the next invocation only.

`\hook_log:n` prints the information to the `.log` file, and `\hook_show:n` prints them to the terminal/command window and starts TeX's prompt (only if `\errorstopmode`) to wait for user action.

The *hook* can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

```
\hook_debug_on: \hook_debug_on:
```

`\hook_debug_off:` Turns the debugging of hook code on or off. This displays changes to the hook data.

2.3 On the order of hook code execution

Chunks of code for a *hook* under different labels are supposed to be independent if there are no special rules set up that define a relation between the chunks. This means that you can't make assumptions about the order of execution!

Suppose you have the following declarations:

```
\NewHook{myhook}
\AddToHook{myhook}{packageA}{\typeout{A}}
\AddToHook{myhook}{packageB}{\typeout{B}}
\AddToHook{myhook}{packageC}{\typeout{C}}
```

then executing the hook with `\UseHook` will produce the typeout A B C in that order. In other words, the execution order is computed to be `packageA`, `packageB`, `packageC` which you can verify with `\ShowHook{myhook}`:

```
-> The hook 'myhook':
> Code chunks:
>     packageA -> \typeout {A}
>     packageB -> \typeout {B}
>     packageC -> \typeout {C}
> Document-level (top-level) code (executed last):
>     ---
> Extra code for next invocation:
>     ---
> Rules:
>     ---
> Execution order:
>     packageA, packageB, packageC.
```

The reason is that the code chunks are internally saved in a property list and the initial order of such a property list is the order in which key-value pairs got added. However, that is only true if nothing other than adding happens!

Suppose, for example, you want to replace the code chunk for `packageA`, e.g.,

```
\RemoveFromHook{myhook}{packageA}  
\AddToHook{myhook}{packageA}{\typeout{A alt}}
```

then your order becomes `packageB`, `packageC`, `packageA` because the label got removed from the property list and then re-added (at its end).

While that may not be too surprising, the execution order is also sometimes altered if you add a redundant rule, e.g. if you specify

```
\DeclareHookRule{myhook}{packageA}{before}{packageB}
```

instead of the previous lines we get

```
-> The hook 'myhook':  
> Code chunks:  
>     packageA -> \typeout {A}  
>     packageB -> \typeout {B}  
>     packageC -> \typeout {C}  
> Document-level (top-level) code (executed last):  
>     ---  
> Extra code for next invocation:  
>     ---  
> Rules:  
>     packageB|packageA with relation >  
> Execution order (after applying rules):  
>     packageA, packageC, packageB.
```

As you can see the code chunks are still in the same order, but in the execution order for the labels `packageB` and `packageC` have swapped places. The reason is that, with the rule there are two orders that satisfy it, and the algorithm for sorting happened to pick a different one compared to the case without rules (where it doesn't run at all as there is nothing to resolve). Incidentally, if we had instead specified the redundant rule

```
\DeclareHookRule{myhook}{packageB}{before}{packageC}
```

the execution order would not have changed.

In summary: it is not possible to rely on the order of execution unless there are rules that partially or fully define the order (in which you can rely on them being fulfilled).

2.4 The use of “reversed” hooks

You may have wondered why you can declare a “reversed” hook with `\NewReversedHook` and what that does exactly.

In short: the execution order of a reversed hook (without any rules!) is exactly reversed to the order you would have gotten for a hook declared with `\NewHook`.

This is helpful if you have a pair of hooks where you expect to see code added that involves grouping, e.g., starting an environment in the first and closing that environment in the second hook. To give a somewhat contrived example⁶, suppose there is a package adding the following:

⁶there are simpler ways to achieve the same effect.

```
\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 saying that `\begin{color}` was 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):
>     ---
> 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}{code-2}` 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.

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.

⁷This is sometimes used with `\AtBeginDocument` which is why it is supported.

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` provides hooks such as `\babel@{language}/afterextras`. However, 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 Hooks with arguments

Sometimes it is necessary to pass contextual information to a hook, and, for one reason or another, it is not feasible to store such information in macros. To serve this purpose, hooks can be declared with arguments, so that the programmer can pass along the data necessary for the code in the hook to function properly.

A hook with arguments works mostly like a regular hook, and most commands that work for regular hooks, also work for hooks that take arguments. The differences are when the hook is declared (`\NewHookWithArguments` is used instead of `\NewHook`), then code can be added with both `\AddToHook` and `\AddToHookWithArguments`, and when the hook is used (`\UseHookWithArguments` instead of `\UseHook`).

A hook with arguments must be declared as such (before it is first used, as all regular hooks) using `\NewHookWithArguments{\langle hook \rangle}{\langle number \rangle}`. All code added to that hook can then use `#1` to access the first argument, `#2` to access the second, and so forth up to the number of arguments declared. However, it is still possible to add code with references to the arguments of a hook that was not yet declared (we will discuss that later). At their core, hooks are macros, so `TeX`'s limit of 9 arguments applies, and a low-level `TeX` error is raised if you try to reference an argument number that doesn't exist.

To use a hook with arguments, just write `\UseHookWithArguments{\langle hook \rangle}{\langle number \rangle}` followed by a braced list of the arguments. For example, if the hook `test` takes three arguments, write:

```
\UseHookWithArguments{test}{3}{arg-1}{arg-2}{arg-3}
```

then, in the `\langle code \rangle` of the hook, all instances of `#1` will be replaced by `arg-1`, `#2` by `arg-2` and so on. If, at the point of usage, the programmer provides more arguments than the hook is declared to take, the excess arguments are simply ignored by the hook. Behaviour is unpredictable⁸ if too few arguments are provided. If the hook isn't declared, `\langle number \rangle` arguments are removed from the input stream.

⁸The hook *will* take the declared number of arguments, and what will happen depends on what was grabbed, and what the hook code does with its arguments.

Adding code to a hook with arguments can be done with `\AddToHookWithArguments` as well as with the regular `\AddToHook`, to achieve different outcomes. The main difference when it comes to adding code to a hook, in this case, is firstly the possibility of accessing a hook's arguments, of course, and second, how parameter tokens (#₆) are treated.

Using `\AddToHook` in a hook that takes arguments will work as it does for all other hooks. This allows a package developer to add arguments to a hook that otherwise had none without having to worry about compatibility. This means that, for example:

```
\AddToHook{test}{\def\foo#1{Hello, #1!}}
```

will define the same macro `\foo` regardless if the hook `test` takes arguments or not.

Using `\AddToHookWithArguments` allows the `\langle code\rangle` added to access the arguments of the hook with #1, #2, and so forth, up to the number of the arguments declared in the hook. This means that if one wants to add a #₆ to the `\langle code\rangle` that token must be doubled in the input. The same definition from above, using `\AddToHookWithArguments`, needs to be rewritten:

```
\AddToHookWithArguments{test}{\def\foo##1{Hello, ##1!}}
```

Extending the above example to use the hook arguments, we could rewrite something like (now from declaration to usage, to get the whole picture):

```
\NewHookWithArguments{test}{1}
\AddToHookWithArguments{test}{%
    \typeout{Defining foo with "#1"}
    \def\foo##1{Hello, ##1! Some text after: #1}%
}
\UseHook{test}{Howdy!}
>ShowCommand\foo
```

Running the code above prints in the terminal:

```
Defining foo with "Howdy!"
> \foo=macro:
#1->Hello, #1! Some text after: Howdy!.
```

Note how ##1 in the call to `\AddToHookWithArguments` became #1, and the #1 was replaced by the argument passed to the hook. Should the hook be used again, with a different argument, the definition would naturally change.

It is possible to add code referencing a hook's arguments before such hook is declared and the number of hooks is fixed. However, if some code is added to the hook, that references more arguments than will be declared for the hook, there will be a low-level TeX error about an “Illegal parameter number” at the time the hook is declared, which will be hard to track down because at that point TeX can't know whence the offending code came from. Thus it is important that package writers explicitly document how many arguments (if any) each hook can take, so users of those packages know how many arguments can be referenced, and equally important, what each argument means.

2.8 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.⁹

2.9 Legacy L^AT_EX 2_E interfaces

L^AT_EX 2_E offered a small number of hooks together with commands to add to them. They are listed here and are retained for backwards compatibility.

With the new hook management, several additional hooks have been added to L^AT_EX and more will follow. See the next section for what is already available.

```
\AtBeginDocument \AtBeginDocument [⟨label⟩] {⟨code⟩}
```

If used without the optional argument `⟨label⟩`, it works essentially like before, i.e., it is adding `⟨code⟩` to the hook `begindocument` (which is executed inside `\begin{document}`). However, all code added this way is labeled with the label `top-level` (see section 2.1.6) if done outside of a package or class or with the package/class name if called inside such a file (see section 2.1.5).

This way one can add 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.

⁹As with everything in T_EX 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!”

```
\AtEndDocument \AtEndDocument [<label>] {<code>}
```

Like `\AtBeginDocument` but for the `enddocument` hook.

The few hooks that existed previously in $\text{\LaTeX} 2\epsilon$ used internally commands such as `\@begindocumenthook` and packages sometimes augmented them directly rather than working through `\AtBeginDocument`. For that reason there is currently support for this, that is, if the system detects that such an internal legacy hook command contains code it adds it to the new hook system under the label `legacy` so that it doesn't get lost.

However, over time the remaining cases of direct usage need updating because in one of the future release of \LaTeX we will turn this legacy support off, as it does unnecessary slow down the processing.

3 $\text{\LaTeX} 2\epsilon$ commands and environments augmented by hooks

In this section we describe the standard hooks that are now offered by \LaTeX , 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 \LaTeX 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 $\langle env \rangle$ has now four associated hooks coming with it:

env/ $\langle env \rangle$ /before This hook is executed as part of `\begin` as the very first action, in particular prior to starting the environment group. Its scope is therefore not restricted by the environment.

env/ $\langle env \rangle$ /begin This hook is executed as part of `\begin` directly in front of the code specific to the environment start (e.g., the second argument of `\newenvironment`). Its scope is the environment body.

env/ $\langle env \rangle$ /end This hook is executed as part of `\end` directly in front of the code specific to the end of the environment (e.g., the third argument of `\newenvironment`).

env/ $\langle env \rangle$ /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/ $\langle env \rangle$ /before` and to `env/ $\langle env \rangle$ /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.¹⁰ 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/ $\langle env \rangle$ /before` hook using the $\langle label \rangle$. If $\langle label \rangle$ is not given, the $\langle default label \rangle$ is used (see section 2.1.5).

\AtBeginEnvironment `\AtBeginEnvironment [<label>] {<env>} {<code>}`

This is like `\BeforeBeginEnvironment` but it adds to the `env/ $\langle env \rangle$ /begin` hook.

\AtEndEnvironment `\AtEndEnvironment [<label>] {<env>} {<code>}`

This is like `\BeforeBeginEnvironment` but it adds to the `env/ $\langle env \rangle$ /end` hook.

\AfterEndEnvironment `\AfterEndEnvironment [<label>] {<env>} {<code>}`

This is like `\BeforeBeginEnvironment` but it adds to the `env/ $\langle env \rangle$ /after` hook.

¹⁰Thus 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 has always provided `\AtEndDocument` to add code to the `\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).

3.7 Hook provided by the mark mechanism

See `lmarks-doc.pdf` for details.

insertmark This hook allows for a special setup while `\InsertMark` inserts a mark. It is executed in group so local changes only apply to the mark being inserted.

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 of 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 of definition for `\hook_debug_on`: and others. These functions are documented on page 206.)

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 of 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 of 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 of 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 of 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 of 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 of 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 of 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 of 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 of definition for _hook_tmp:w.)

\c_hook_empty_tl An empty token list, and one containing nine parameters.

35 \tl_const:Nn \c_hook_empty_tl { }
36 \tl_const:Nn \c_hook_nine_parameters_tl { #1#2#3#4#5#6#7#8#9 }

(End of definition for \c_hook_empty_tl and \c_hook_nine_parameters_tl.)

\tl_gremove_once:Nx Some variants of expl3 functions.

\tl_show:x
\tl_log:x

\tl_set:N

\cs_replacement_spec:c

\prop_put:Nne

\str_count:e

37 \cs_generate_variant:Nn \tl_gremove_once:Nn { Nx }
38 \cs_generate_variant:Nn \tl_show:n { x }
39 \cs_generate_variant:Nn \tl_log:n { x }
40 \cs_generate_variant:Nn \tl_set:Nn { Ne }
41 \cs_generate_variant:Nn \cs_replacement_spec:N { c }
42 \cs_generate_variant:Nn \prop_put:Nnn { Nne }
43 \cs_generate_variant:Nn \str_count:n { e }

(End of definition for \tl_gremove_once:Nx and others.)

\s_hook_mark Scan mark used for delimited arguments.

44 \scan_new:N \s_hook_mark

(End of definition for \s_hook_mark.)

_hook_use_none_delimit_by_s_mark:w Removes tokens until the next \s_hook_mark.

45 \cs_new:Npn _hook_use_none_delimit_by_s_mark:w #1 \s_hook_mark { }
46 \cs_new:Npn _hook_use_i_delimit_by_s_mark:nw #1 #2 \s_hook_mark {#1}

(End of definition for _hook_use_none_delimit_by_s_mark:w and
_hook_use_i_delimit_by_s_mark:nw.)

_hook_tl_set:cn 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.

47 \cs_new_protected:Npn _hook_tl_set:cn #1#2
48 { \cs_set_nopar:cp {#1} { _kernel_exp_not:w {#2} } }

(End of definition for _hook_tl_set:cn.)

```

\__hook_tl_gset:Nn  Same as above.
\__hook_tl_gset:Nx  49 \cs_new_protected:Npn \__hook_tl_gset:Nn #1#2
\__hook_tl_gset:cn  50 { \cs_gset_nopar:Npx #1 { \__kernel_exp_not:w {#2} } }
\__hook_tl_gset:co  51 \cs_new_protected:Npn \__hook_tl_gset:Nx #1#2
\__hook_tl_gset:cx  52 { \cs_gset_nopar:Npx #1 {#2} }
53 \cs_generate_variant:Nn \__hook_tl_gset:Nn { c, co }
54 \cs_generate_variant:Nn \__hook_tl_gset:Nx { c }

(End of definition for \__hook_tl_gset:Nn.)

\__hook_tl_gput_right:Nn  Same as above.
\__hook_tl_gput_right:Ne  55 \cs_new_protected:Npn \__hook_tl_gput_right:Nn #1#2
\__hook_tl_gput_right:cn  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 { Ne, cn }

(End of definition for \__hook_tl_gput_right:Nn.)

\__hook_tl_gput_left:Nn  Same as above.
58 \cs_new_protected:Npn \__hook_tl_gput_left:Nn #1#2
59 {
60     \__hook_tl_gset:Nx #1
61     { \__kernel_exp_not:w {#2} \__kernel_exp_not:w \exp_after:wN {#1} }
62 }

(End of definition for \__hook_tl_gput_left:Nn.)

\__hook_tl_gset_eq:NN  Same as above.
63 \cs_new_eq:NN \__hook_tl_gset_eq:NN \tl_gset_eq:NN

(End of definition for \__hook_tl_gset_eq:NN.)

\__hook_tl_gclear:N  Same as above.
\__hook_tl_gclear:c  64 \cs_new_protected:Npn \__hook_tl_gclear:N #1
65 { \__hook_tl_gset_eq:NN #1 \c_empty_tl }
66 \cs_generate_variant:Nn \__hook_tl_gclear:N { c }

(End of definition for \__hook_tl_gclear:N.)

```

4.4 Providing new hooks

4.4.1 The data structures of a hook

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

\g_@@_{hook}_reversed_tl \g_{hook}_{hook}_code_prop A property list holding the code for the hook in separate
\g_@@_{hook}_declared_tl chunks. The keys are by default the package names that add code to the hook, but
\g_@@_{hook}_parameter_tl it is possible for packages to define other keys.

\@@_{next}_{hook} \g_{hook}_{hook}_rule_{label1}|label2_tl A token list holding the relation between *(label1)* and *(label2)* in the *<hook>*. The *(labels)* are lexically (reverse) sorted to ensure that two labels always point to the same token list. For global rules, the *(hook)* name is ??.

`__hook_{hook}` The code that is actually executed when the hook is called in the document is stored in this token list. It is constructed from the code chunks applying the information. This token list is named like that so that in case of an error inside the hook, the reported token list in the error is shorter, and to make it simpler to normalize hook names in `_hook_make_name:n`.

`\g_hook_{hook}_reversed_t1` Some hooks are “reversed”. This token list stores a `-` for such hook so that it can be identified. The `-` character is used because `\{reversed\}1` is `+1` for normal hooks and `-1` for reversed ones.

`\g_hook_{hook}_declared_t1` This token list serves as a marker for the hook being officially declared. Its existence is tested to raise an error in case another declaration is attempted.

`\c_hook_{hook}_parameter_t1` This token list stores the parameter text for a declared hook (its existence almost completely intersects the token list above), which is used for managing hooks with arguments.

`__hook_toplevel_{hook}` This token list stores the code inserted in the hook from the user’s document, in the `top-level` label. This label is special, and doesn’t participate in sorting. Instead, all code is appended to it and executed after (or before, if the hook is reversed) the normal hook code, but before the `next` code chunk.

`__hook_next_{hook}` Finally there is extra code (normally empty) that is used on the next invocation of the hook (and then deleted). This can be used to define some special behavior for a single occasion from within the document. This token list follows the same naming scheme than the main `__hook_{hook}` token list. It is called `__hook_next_{hook}` rather than `__hook_{next_{hook}}` because otherwise a hook whose name is `next_{hook}` would clash with the next code-token list of the hook called `{hook}`.

4.4.2 On the existence of hooks

A hook may be in different states of existence. Here we give an overview of the internal commands to set up hooks and explain how the different states are distinguished. The actual implementation then follows in subsequent sections.

One problem we have to solve is that we need to be able to add code to hooks (e.g., with `\AddToHook`) even if that code has not yet been declared. For example, one package needs to write into a hook of another package, but that package may not get loaded, or is loaded only later. Another problem is that most hooks, but not the generic hooks, require a declaration.

We therefore distinguish the following states for a hook, which are managed by four different tests: structure existence (`__hook_if_structure_exist:nTF`), creation (`__hook_if_usable:nTF`), declaration (`__hook_if_declared:nTF`) and disabled or not (`__hook_if_disabled:nTF`)

not existing Nothing is known about the hook so far. This state can be detected with `__hook_if_structure_exist:nTF` (which uses the false branch).

In this state the hook can be declared, disabled, rules can be defined or code could be added to it, but it is not possible to use the hook (with `\UseHook`).

basic data structure set up A hook is in this state when its basic data structure has been set up (using `__hook_init_structure:n`). The data structure setup happens automatically when commands such as `\AddToHook` are used and the hook is at that point in state “not existing”.

In this state the four tests give the following results:

```
\__hook_if_structure_exist:nTF returns true.  
    \__hook_if_usable:nTF returns false.  
    \__hook_if_declared:nTF returns false.  
    \__hook_if_disabled:nTF returns false.
```

The allowed actions are the same as in the “not existing” state.

declared A hook is in this state 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., `\begin{document}`.

```

--hook_new:nn
 67 <|latexrelease>\IncludeInRelease{2023/06/01}{\hook_new_with_args:nn}
 68 <|latexrelease>                                {Hooks~with~args}
 69 \cs_new_protected:Npn \hook_new:nn #1
 70   { \__hook_normalize_hook_args:Nn \__hook_new:nn {#1} { 0 } }
 71 \cs_new_protected:Npn \hook_new_with_args:nn #1 #2
 72   { \__hook_normalize_hook_args:Nn \__hook_new:nn {#1} {#2} }
 73 \cs_new_protected:Npn \__hook_new:nn #1 #2
 74   {

```

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.

```

75   \__hook_if_declared:nTF {#1}
76     { \msg_error:nnn { hooks } { exists } {#1} }
77     {
78       \tl_new:c { g__hook_#1_declared_tl }
79       \cs_undefine:c { __hook~#1 }
80       \cs_undefine:c { c__hook_#1_parameter_tl }
81       \__hook_make_usable:nn {#1} {#2}

```

In case there is already code in a hook, but it’s undeclared, run `__hook_update_hook_code:n` to make it ready to be executed (see test `lthooks-034`).

```

82     \__hook_update_hook_code:n {#1}
83   }
84 }
85 <|latexrelease>\EndIncludeInRelease
86 <|latexrelease>\IncludeInRelease{2020/10/01}{\hook_new_with_args:nn}
87 <|latexrelease>                                {Hooks~with~args}
88 <|latexrelease>\cs_gset_protected:Npn \hook_new:n #1
89 <|latexrelease>  { \__hook_normalize_hook_args:Nn \__hook_new:n {#1} }
90 <|latexrelease>\cs_undefine:N \__hook_new:nn
91 <|latexrelease>\cs_gset_protected:Npn \__hook_new:n #1
92 <|latexrelease>  {
93 <|latexrelease>    \__hook_if_declared:nTF {#1}
94 <|latexrelease>      { \msg_error:nnn { hooks } { exists } {#1} }
95 <|latexrelease>      {
96 <|latexrelease>        \tl_new:c { g__hook_#1_declared_tl }
97 <|latexrelease>        \__hook_make_usable:n {#1}
98 <|latexrelease>      }
99 <|latexrelease>  }
100 <|latexrelease>\cs_gset_protected:Npn \hook_new_with_args:nn #1 { }
101 <|latexrelease>\EndIncludeInRelease

```

(End of definition for `\hook_new:n`, `\hook_new_with_args:nn`, and `__hook_new:nn`. These functions are documented on page 203.)

`__hook_make_usable:nn` 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:nnn` when adding code to a generic hook.

```

102  \if latexrelease \IncludeInRelease{2023/06/01}{\_hook_make_usable:nn}
103  \if latexrelease
104    \cs_new_protected:Npn \_hook_make_usable:nn #1 #2
105    {

```

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.

```

106    \_hook_if_usable:nF {#1}
107    {
108      \seq_gput_right:Nn \g__hook_all_seq {#1}

```

Here we'll define the `\c__hook_<hook>_parameter_tl` to hold a run of parameters up to the number of arguments of the hook (#2).

```

109  \_kernel_cs_parm_from_arg_count:nnF
110  {
111    \tl_const:cN { c__hook_#1_parameter_tl } } {#2}
112    \msg_error:nnnn { hooks } { too-many-args } {#1} {#2}
113    \tl_const:cX { c__hook_#1_parameter_tl }
114    { \exp_not:V \c__hook_nine_parameters_tl }
115  }

```

After that, use `_hook_normalise_cs_args:nn` to correct the number of parameters of the macros `_hook_toplevel_<hook>` and `_hook_next_<hook>`. We need to be able to add code with arguments to a hook without prior knowledge of the number of arguments of that hook, so `lthooks` assumes 9 until the hook is properly declared and the number of arguments is known. `_hook_normalise_cs_args:nn` does the normalisation by using the `\c__hook_<hook>_parameter_tl` defined just above.

```

116  \_hook_normalise_cs_args:nn { _toplevel } {#1}
117  \_hook_normalise_cs_args:nn { _next } {#1}

```

This is only used by the actual code of the current hook, so declare it normally:

```

118  \_hook_code_gset:nn {#1} { }

```

Now ensure that the base data structure for the hook exists:

```

119  \_hook_init_structure:n {#1}

```

The call to `_hook_normalise_code_pool:n` will correct any improper reference to arguments that don't exist in the hook, raising a low-level `TeX` error and doubling the offending parameter tokens. It has to be done after `_hook_init_structure:n` because it operates on `\g__hook_<hook>_code_prop`.

```

120  \_hook_normalise_code_pool:n {#1}

```

The `\g__hook_<hook>_labels_clist` holds the sorted list of labels (once it got sorted). This is used only for debugging. These are defined conditionally, in case `_hook_make_usable:nn` is being used to redefine a hook.

```

121  \clist_if_exist:cF { g__hook_#1_labels_clist }
122  {
123    \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.

```

124  \tl_new:c { g__hook_#1_reversed_tl }
125  }

```

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., $\text{\@begindocumenthook}$. 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., $\text{\@hook-name}hook$ 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!

```

126      \_\_hook\_include\_legacy\_code\_chunk:n {#1}
127      }
128  }
129  \EndIncludeInRelease
130 \IncludeInRelease{2020/10/01}{\_\_hook\_make\_usable:nn}
131 {Hooks~with-args}
132 \cs_undefine:N \_\_hook\_make\_usable:nn
133 \cs_gset_protected:Npn \_\_hook\_make\_usable:n #1
134 \tl_if_exist:cF { \_\_hook~#1 }
135 \tl_new:c { \_\_hook~#1 }
136 \_\_hook_init_structure:n {#1}
137 \seq_gput_right:Nn \g_\_\_hook_all_seq {#1}
138 \tl_new:c { \_\_hook~#1 }
139 \_\_hook_init_structure:n {#1}
140 \clist_new:c { g_\_\_hook_#1_labels_clist }
141 \tl_new:c { g_\_\_hook_#1_reversed_tl }
142 \_\_hook_include_legacy_code_chunk:n {#1}
143 \tl_new:c { \_\_hook~#1 }
144 \EndIncludeInRelease
145 \EndIncludeInRelease

```

(End of definition for `__hook_make_usable:nn`.)

`__hook_init_structure:n` This function declares the basic data structures for a hook without explicitly 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`.

```

146 \IncludeInRelease{2023/06/01}{\_\_hook_init_structure:n}
147 {Hooks~with-args}
148 \cs_new_protected:Npn \_\_hook_init_structure:n #1
149 {
150   \_\_hook_if_structure_exist:nF {#1}
151   {
152     \prop_new:c { g_\_\_hook_#1_code_prop }
153     \_\_hook_toplevel_gset:nn {#1} { }
154     \_\_hook_next_gset:nn {#1} { }
155   }
156 }
157 \EndIncludeInRelease
158 \IncludeInRelease{2020/10/01}{\_\_hook_init_structure:n}
159 {Hooks~with-args}

```

```

160 〈latexrelease〉\cs_gset_protected:Npn \__hook_init_structure:n #1
161 〈latexrelease〉  {
162 〈latexrelease〉    \__hook_if_structure_exist:nF {#1}
163 〈latexrelease〉    {
164 〈latexrelease〉      \prop_new:c { g__hook_#1_code_prop }
165 〈latexrelease〉      \tl_new:c { __hook_toplevel~#1 }
166 〈latexrelease〉      \tl_new:c { __hook_next~#1 }
167 〈latexrelease〉    }
168 〈latexrelease〉  }
169 〈latexrelease〉\EndIncludeInRelease

```

(End of definition for `__hook_init_structure:n`.)

\hook_new_reversed:n
\hook_new_reversed_with_args:nn

Declare a new hook. The default ordering of code chunks is reversed, signaled by setting the token list to a minus sign.

```

\__hook_new_reversed:nn
170 〈latexrelease〉\IncludeInRelease{2023/06/01}{\hook_new_reversed_with_args:nn}
171 〈latexrelease〉
172 \cs_new_protected:Npn \hook_new_reversed:n #1
173   { \__hook_normalize_hook_args:Nn \__hook_new_reversed:nn {#1} { 0 } }
174 \cs_new_protected:Npn \hook_new_reversed_with_args:nn #1 #2
175   { \__hook_normalize_hook_args:Nn \__hook_new_reversed:nn {#1} {#2} }
176 \cs_new_protected:Npn \__hook_new_reversed:nn #1 #2
177   {
178     \__hook_if_declared:nTF {#1}
179       { \msg_error:nnn { hooks } { exists } {#1} }
180       {
181         \__hook_new:nn {#1} {#2}
182         \tl_gset:cn { g__hook_#1_reversed_tl } { - }
183       }
184   }
185 〈latexrelease〉\EndIncludeInRelease

186 〈latexrelease〉\IncludeInRelease{2020/10/01}{\hook_new_reversed_with_args:nn}
187 〈latexrelease〉
188 \cs_gset_protected:Npn \hook_new_reversed:n #1
189 〈latexrelease〉  { \__hook_normalize_hook_args:Nn \__hook_new_reversed:n {#1} }
190 \cs_undefine:N \__hook_new_reversed:nn
191 \cs_gset_protected:Npn \__hook_new_reversed:n #1
192 〈latexrelease〉  {
193 〈latexrelease〉    \__hook_new:n {#1}
194 〈latexrelease〉    \tl_gset:cn { g__hook_#1_reversed_tl } { - }
195 〈latexrelease〉  }
196 \cs_undefine:N \__hook_new_reversed:nn
197 \cs_gset_protected:Npn \hook_new_reversed_with_args:nn #1 #2 { }
198 〈latexrelease〉\EndIncludeInRelease

```

(End of definition for `\hook_new_reversed:n`, `\hook_new_reversed_with_args:nn`, and `__hook_new_reversed:nn`. These functions are documented on page 203.)

\hook_new_pair:nn
\hook_new_pair_with_args:nnn

A shorthand for declaring a normal and a (matching) reversed hook in one go.

```

199 〈latexrelease〉\IncludeInRelease{2023/06/01}{\hook_new_pair_with_args:nnn}
200 〈latexrelease〉
201 \cs_new_protected:Npn \hook_new_pair:nn #1#2
202   { \__hook_normalize_hook_args:Nnn \__hook_new_pair:nnn {#1} {#2} { 0 } }
203 \cs_new_protected:Npn \hook_new_pair_with_args:nnn #1#2#3

```

```

204 { \__hook_normalize_hook_args:Nnn \__hook_new_pair:nnn {#1} {#2} {#3} }
205 \cs_new_protected:Npn \__hook_new_pair:nnn #1 #2 #3
206 {
207     \__hook_if_declared:nTF {#1}
208     { \msg_error:nnn { hooks } { exists } {#1} }
209     {
210         \__hook_if_declared:nTF {#2}
211         { \msg_error:nnn { hooks } { exists } {#2} }
212         {
213             \__hook_new:nn {#1} {#3}
214             \__hook_new_reversed:nn {#2} {#3}
215         }
216     }
217 }
218 \end{IncludeInRelease}
219 \IncludeInRelease{2020/10/01}{\hook_new_pair_with_args:nnn}
220 \begin{IncludeInRelease}
221 \cs_gset_protected:Npn \hook_new_pair:nn #1#2
222 \begin{IncludeInRelease}
223 \hook_new:n {#1}
224 \hook_new_reversed:n {#2}
225 \end{IncludeInRelease}
226 \cs_gset_protected:Npn \hook_new_pair_with_args:nnn #1#2#3
227 \end{IncludeInRelease}
228 \end{IncludeInRelease}

```

(End of definition for `\hook_new_pair:nn` and `\hook_new_pair_with_args:nnn`. These functions are documented on page 203.)

`__hook_include_legacy_code_chunk:`

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!

```

229 \IncludeInRelease{2023/06/01}{\__hook_include_legacy_code_chunk:n}
230 \begin{IncludeInRelease}
231 \cs_new_protected:Npn \__hook_include_legacy_code_chunk:n #1
232 {

```

If the macro doesn't exist (which is the usual case) then nothing needs to be done.

```

233 \tl_if_exist:cT { @#1hook }
234 {

```

Of course if the legacy hook exists but is empty, there is no need to add anything under `legacy` the legacy label.

```

235 \tl_if_empty:cF { @#1hook }
236 {

```

Here we set `__hook_replacing_args_false:` because no legacy code will reference hook arguments.

```

237 \__hook_replacing_args_false:
238 \use:e

```

```

239      {
240          \__hook_hook_gput_code_do:nnn {#1} { legacy }
241          { \exp_not:v { @#1hook } }
242      }
243      \__hook_replacing_args_reset:
244          \__hook_tl_gclear:c { @#1hook }
245      }
246  }
247 }
248 <|latexrelease>\EndIncludeInRelease
249 <|latexrelease>\IncludeInRelease{2020/10/01}{\__hook_include_legacy_code_chunk:n}
250 <|latexrelease>                      {Hooks~with~args}
251 <|latexrelease>\cs_gset_protected:Npn \__hook_include_legacy_code_chunk:n #1
252 <|latexrelease>  {
253 <|latexrelease>      \tl_if_exist:cT { @#1hook }
254 <|latexrelease>      {
255 <|latexrelease>          \tl_if_empty:cF { @#1hook }
256 <|latexrelease>          {
257 <|latexrelease>              \exp_args:Nnnv \__hook_hook_gput_code_do:nnn
258 <|latexrelease>                  {#1} { legacy } { @#1hook }
259 <|latexrelease>              \__hook_tl_gclear:c { @#1hook }
260 <|latexrelease>          }
261 <|latexrelease>      }
262 <|latexrelease>  }
263 <|latexrelease>\EndIncludeInRelease

```

(End of definition for `__hook_include_legacy_code_chunk:n`.)

4.4.4 Disabling and providing hooks

\hook_disable_generic:n 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.

__hook_disable:n 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.

```

264 <|latexrelease>\IncludeInRelease{2021/06/01}{\hook_disable_generic:n}
265 <|latexrelease>                      {Disable~hooks}
266 \cs_new_protected:Npn \hook_disable_generic:n #1
267     { \__hook_normalize_hook_args:Nn \__hook_disable:n {#1} }
268 \cs_new_protected:Npn \__hook_disable:n #1
269  {
270      \tl_gclear_new:c { g__hook_#1_declared_tl }
271      \cs_undefine:c { __hook~#1 }
272  }
273 \prg_new_conditional:Npnn \__hook_if_disabled:n #1 { p, T, F, TF }
274  {
275      \bool_lazy_and:nnTF
276          { \tl_if_exist_p:c { g__hook_#1_declared_tl } }
277          { ! \cs_if_exist_p:c { __hook~#1 } }

```

```

278      { \prg_return_true: }
279      { \prg_return_false: }
280  }
281 <latexrelease>\EndIncludeInRelease
282 <latexrelease>\IncludeInRelease{2020/10/01}{\hook_disable_generic:n}
283 <latexrelease>                                {Disable~hooks}
284 <latexrelease>
285 <latexrelease>\cs_new_protected:Npn \hook_disable_generic:n #1 {}
286 <latexrelease>
287 <latexrelease>\EndIncludeInRelease

(End of definition for \hook_disable_generic:n, \_\_hook_disable:n, and \_\_hook_if_disabled:nTF.
This function is documented on page 204.)
```

\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.

```

288 <latexrelease>\IncludeInRelease{2023/06/01}{\hook_activate_generic:n}
289 <latexrelease>                                {Providing~hooks}
290 \cs_new_protected:Npn \hook_activate_generic:n #1
291   { \_\_hook_normalize_hook_args:Nn \_\_hook_activate_generic:nn {#1} { } }
292 \cs_new_protected:Npn \_\_hook_activate_generic:nn #1 #2
293   { }
```

If the hook to be activated was disabled we warn (for now — this may change).

```

294   \_\_hook_if_disabled:nTF {#1}
295     { \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.

```

296   {
297     \_\_hook_if_declared:nF {#1}
298     {
299       \tl_new:c { g\_hook_#1_declared_tl }
300       \_\_hook_make_usable:nn {#1} { 0 }
301       \tl_gset:cx { g\_hook_#1_reversed_tl }
302         { \_\_hook_if_generic_reversed:nT {#1} { - } }
```

Reflect that we have activated the generic hook and set its execution code.

```

303   \_\_hook_update_hook_code:n {#1}
304   }
305   }
306 }

307 <latexrelease>\EndIncludeInRelease
308 <latexrelease>\IncludeInRelease{2021/06/01}{\hook_activate_generic:n}
309 <latexrelease>                                {Providing~hooks}
310 <latexrelease>\cs_gset_protected:Npn \_\_hook_activate_generic:nn #1 #2
311 <latexrelease>  {
312 <latexrelease>    \_\_hook_if_disabled:nTF {#1}
313 <latexrelease>      { \msg_warning:nnn { hooks } { activate-disabled } {#1} }
314 <latexrelease>      {
315 <latexrelease>        \_\_hook_if_declared:nF {#1}
316 <latexrelease>          { }
```

```

317 <|latexrelease>          \tl_new:c { g__hook_##1_declared_tl }
318 <|latexrelease>          \__hook_make_usable:n {#1}
319 <|latexrelease>          \tl_gset:cx { g__hook_##1_reversed_tl }
320 <|latexrelease>          { \__hook_if_generic_reversed:nT {#1} { - } }
321 <|latexrelease>          \__hook_update_hook_code:n {#1}
322 <|latexrelease>          }
323 <|latexrelease>          }
324 <|latexrelease>          }
325 <|latexrelease>\EndIncludeInRelease

326 <|latexrelease>\IncludeInRelease{2020/10/01}{\hook_activate_generic:n}
327 <|latexrelease>                      {Providing-hooks}
328 <|latexrelease>\cs_gset_protected:Npn \hook_activate_generic:n #1 { }
329 <|latexrelease>\EndIncludeInRelease

(End of definition for \hook_activate_generic:n and \__hook_activate_generic:n. This function is
documented on page 204.)

```

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::

```

330 \cs_new:Npn \__hook_parse_label_default:n #1
331 {
332     \tl_if_novalue:nTF {#1}
333     { \__hook_currname_or_default: }
334     { \tl_trim_spaces_apply:nN {#1} \__hook_parse_dot_label:n }
335 }

```

(End of 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.

```

336 \cs_new:Npn \__hook_parse_dot_label:n #1
337 {
338     \tl_if_empty:nTF {#1}
339     {
340         \msg_expandable_error:nn { hooks } { empty-label }
341         \__hook_currname_or_default:
342     }
343     {
344         \str_if_eq:nnTF {#1} { .. }
345         { \__hook_currname_or_default: }
346         { \__hook_parse_dot_label:w #1 ./ \s__hook_mark }
347     }
348 }
349 \cs_new:Npn \__hook_parse_dot_label:w #1 ./ #2 \s__hook_mark
350 {
351     \tl_if_empty:nTF {#1}
352     { \__hook_parse_dot_label_aux:w #2 \s__hook_mark }
353     {
354         \tl_if_empty:nTF {#2}
355         { \__hook_make_name:n {#1} }

```

```

356         { \_\_hook\_parse\_dot\_label\_cleanup:w #1 ./ #2 \s\_hook\_mark }
357     }
358   }
359 \cs_new:Npn \_\_hook\_parse\_dot\_label\_cleanup:w #1 ./ \s\_hook\_mark {#1}
360 \cs_new:Npn \_\_hook\_parse\_dot\_label\_aux:w #1 ./ \s\_hook\_mark
361   { \_\_hook\_currname\_or\_default: / \_\_hook\_make\_name:n {#1} }

(End of definition for \_\_hook\_parse\_dot\_label:n and others.)

```

__hook_currname_or_default: This uses \g_hook_hook_curr_name_tl if it is set, otherwise it tries \currname. If neither is set, it raises an error and uses the fallback value label-missing.

```

362 \cs_new:Npn \_\_hook\_currname\_or\_default:
363   {
364     \tl_if_empty:NTF \g\_hook\_curr\_name\_tl
365     {
366       \tl_if_empty:NTF \currname
367       {
368         \msg_expandable_error:nnn { latex2e } { should-not-happen }
369         { Empty~default-label. }
370         \_\_hook\_make\_name:n { label-missing }
371       }
372       { \currname }
373     }
374     { \g\_hook\_curr\_name\_tl }
375   }

(End of definition for \_\_hook\_currname\_or\_default::)

```

__hook_make_name:n This provides a standard sanitization of a hook's name. It uses \cs:w to build a control sequence out of the hook name, then uses \cs_to_str:N to get the string representation of that, without the escape character. \cs:w-based expansion is used instead of e-based because Unicode characters don't behave well inside \expanded. The macro adds the __hook_ prefix to the hook name to reuse the hook's code token list to build the csname and avoid leaving "public" control sequences defined (as \relax) in TeX's memory.

```

376 \cs_new:Npn \_\_hook\_make\_name:n #1
377   {
378     \exp_after:wN \exp_after:wn \exp_after:wN \_\_hook\_make\_name:w
379     \exp_after:wN \token_to_str:N \cs:w \_\_hook~ #1 \cs_end:
380   }
381 \exp_last_unbraced:NNNNo
382 \cs_new:Npn \_\_hook\_make\_name:w #1 \tl_to_str:n { \_\_hook~ } { }

(End of definition for \_\_hook\_make\_name:n and \_\_hook\_make\_name:w.)

```

__hook_normalize_hook_args:Nn This is the standard route for normalizing hook and label arguments. The main macro does the entire operation within a group so that csnames made by __hook_make_name:n are wiped off before continuing. This means that this function cannot be used for \hook_use:n!

```

383 \cs_new_protected:Npn \_\_hook\_normalize\_hook\_args_aux:Nn #1 #2
384   {
385     \group_begin:
386     \use:e
387     {
388       \group_end:

```

```

389           \exp_not:N #1 #2
390       }
391   }
392 \cs_new_protected:Npn \__hook_normalize_hook_args:Nn #1 #2
393 {
394     \__hook_normalize_hook_args_aux:Nn #1
395     { { \__hook_parse_label_default:n {#2} } }
396   }
397 \cs_new_protected:Npn \__hook_normalize_hook_args:Nnn #1 #2 #3
398 {
399     \__hook_normalize_hook_args_aux:Nn #1
400     {
401         { \__hook_parse_label_default:n {#2} }
402         { \__hook_parse_label_default:n {#3} }
403     }
404   }
405 \cs_new_protected:Npn \__hook_normalize_hook_rule_args:Nnnnn #1 #2 #3 #4 #5
406 {
407     \__hook_normalize_hook_args_aux:Nn #1
408     {
409         { \__hook_parse_label_default:n {#2} }
410         { \__hook_parse_label_default:n {#3} }
411         { \tl_trim_spaces:n {#4} }
412         { \__hook_parse_label_default:n {#5} }
413     }
414 }

```

(End of 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`:

```
415 \tl_gset:Nn \g__hook_hook_curr_name_tl { top-level }
```

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

```
416 <\textralatexrelease> \seq_if_empty:NT \g__hook_name_stack_seq
417 <\textralatexrelease> { \seq_gput_right:Nn \g__hook_name_stack_seq { top-level } }
```

then we dissect the `\@currnamestack`, adding `\@currname` to the stack:

```
418 <\textralatexrelease> \cs_set_protected:Npn \__hook_tmp:w #1 #2 #3
419 <\textralatexrelease> {
420 <\textralatexrelease>     \quark_if_recursion_tail_stop:n {#1}
421 <\textralatexrelease>     \seq_gput_right:Nn \g__hook_name_stack_seq {#1}
422 <\textralatexrelease>     \__hook_tmp:w
423 <\textralatexrelease> }
424 <\textralatexrelease> \exp_after:wN \__hook_tmp:w \@currnamestack
425 <\textralatexrelease> \q_recursion_tail \q_recursion_tail
426 <\textralatexrelease> \q_recursion_tail \q_recursion_stop
```

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

```
427 \t1_gset:Nx \g__hook_hook_curr_name_tl { \@currname }
428 \seq_gpop_right>NN \g__hook_name_stack_seq \l__hook_tmpa_tl
```

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):

```
429 \cs_new_protected:Npn \__hook_curr_name_push:n #1
430   { \exp_args:Nx \__hook_curr_name_push_aux:n { \__hook_make_name:n {#1} } }
431 \cs_new_protected:Npn \__hook_curr_name_push_aux:n #1
432   {
433     \tl_if_blank:nTF {#1}
434       { \msg_error:nn { hooks } { no-default-label } }
435       {
436         \str_if_eq:nnTF {#1} { top-level }
437           {
438             \msg_error:nnnn { hooks } { set-top-level }
439               { to } { PushDefaultHookLabel } {#1}
440           }
441           {
442             \seq_gpush:NV \g__hook_name_stack_seq \g__hook_hook_curr_name_tl
443             \t1_gset:Nn \g__hook_hook_curr_name_tl {#1}
444           }
445       }
446   }
```

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:

```
447 \cs_new_protected:Npn \__hook_curr_name_pop:
448   {
449     \seq_gpop:NNTF \g__hook_name_stack_seq \l__hook_return_tl
450       { \t1_gset_eq:NN \g__hook_hook_curr_name_tl \l__hook_return_tl }
451       { \msg_error:nn { hooks } { extra-pop-label } }
452   }
```

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):

```
453 \t1_gput_right:Nn \@kernel@after@enddocument@afterlastpage
454   { \__hook_end_document_label_check: }
455 \cs_new_protected:Npn \__hook_end_document_label_check:
456   {
457     \seq_gpop:NNT \g__hook_name_stack_seq \l__hook_return_tl
458     {
459       \msg_error:nnx { hooks } { missing-pop-label }
460         { \g__hook_hook_curr_name_tl }
461       \t1_gset_eq:NN \g__hook_hook_curr_name_tl \l__hook_return_tl
462         \__hook_end_document_label_check:
463     }
464 }
```

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.

```

465 \cs_new_protected:Npn \__hook_set_default_hook_label:n #1
466 {
467     \seq_if_empty:NTF \g__hook_name_stack_seq
468     {
469         \msg_error:nnnn { hooks } { set-top-level }
470         { for } { SetDefaultHookLabel } {#1}
471     }
472     { \exp_args:Nx \__hook_set_default_label:n { \__hook_make_name:n {#1} } }
473 }
474 \cs_new_protected:Npn \__hook_set_default_label:n #1
475 {
476     \str_if_eq:nnTF {#1} { top-level }
477     {
478         \msg_error:nnnn { hooks } { set-top-level }
479         { to } { SetDefaultHookLabel } {#1}
480     }
481     { \tl_gset:Nn \g__hook_hook_curr_name_tl {#1} }
482 }
```

(End of definition for `__hook_curr_name_push:n` and others.)

4.6 Adding or removing hook code

With `\hook_gput_code:nnn{⟨hook⟩}{⟨label⟩}{⟨code⟩}` a chunk of `⟨code⟩` is added to an existing `⟨hook⟩` labeled with `⟨label⟩`.

```

\hook_gput_code:nnn
\hook_gput_code_with_args:nnn
\__hook_gput_code:nnn
\__hook_gput_code_store:nnn
\__hook_gput_code_do:nnn
\__hook_prop_gput_labeled_cleanup:nnn
\__hook_prop_gput_labeled_do:Nnn
\__hook_replacing_args_false:
\__hook_normalize_hook_args:Nnn \__hook_gput_code:nnn {#1} {#2} {#3}
\__hook_replacing_args_reset:
\__hook_replacing_args_true:
\__hook_normalize_hook_args:Nnn \__hook_gput_code:nnn {#1} {#2} {#3}
\__hook_replacing_args_reset:
```

If `\AddToHookWithArguments` was used, do some sanity checking, and if it's not possible to use arguments at this point, fall back to regular `\AddToHook` by using `__hook_replacing_args_false`:

```

\cs_new_protected:Npn \__hook_gput_code:nnn #1 #2 #3
{
    \__hook_chk_args_allowed:nn {#1} { AddToHook }
```

Then check if the code should be executed immediately, rather than stored:

```

\__hook_if_execute_immediately:nTF {#1}
{

```

\AddToHookWithArguments can't be used on one-time hooks (that were already used).

```
502     \_\_hook\_if\_replacing\_args:TF
503     {
504         \msg_error:nnn { hooks } { one-time-args }
505         {#1} { AddToHook }
506     }
507     {
508         \use:n
509     }
510     { \_\_hook_gput_code_store:nnn {#1} {#2} }
511     {#3}
512 }
513 \cs_new_protected:Npn \_\_hook_gput_code_store:nnn #1 #2 #3
514 {
```

Then check if the hook is usable.

```
515     \_\_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.

```
516     {
517         \_\_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.

```
518     \_\_hook_update_hook_code:n {#1}
519 }
```

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.

```
520     {
521         \_\_hook_if_disabled:nTF {#1}
522         { \msg_error:nn { hooks } { hook-disabled } {#1} }
523         { \_\_hook_try_declarng_generic_hook:nnn {#1} {#2} {#3} }
524     }
525 }
```

This macro will unconditionally add a chunk of code to the given hook.

```
526 \cs_new_protected:Npn \_\_hook_hook_gput_code_do:nnn #1 #2 #3
527 {
```

However, first some debugging info if debugging is enabled:

```
528     \_\_hook_debug:n{\iow_term:x{****~ Add~ to~
529                             \_\_hook_if_usable:nF {#1} { undeclared~ }
530                             hook~ #1~ (#2)
531                             \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}.

```
532     \str_if_eq:nnTF {#2} { top-level }
533     {
534         \str_if_eq:eeTF { top-level } { \_\_hook_currname_or_default: }
535         {
```

If the hook's basic structure does not exist, we need to declare it with `__hook_init_structure:n`.

```
536           \_\_hook\_init\_structure:n {#1}
```

Then append to the `_toplevel` container for the hook.

```
537           \_\_hook\_cs_gput_right:nnn { _toplevel } {#1} {#3}
538       }
539   { \msg_error:nnn { hooks } { misused-top-level } {#1} }
540 }
541 {
```

When adding to the code pool, we have to double hashes if `\AddToHook` was used (`replacing_args` is false), so that later it is turned into a single parameter token, rather than a parameter to the hook macro.

```
542     \exp_args:Nx \_\_hook_prop_gput_labeled_cleanup:nnn
543     {
544         \_\_hook_if_replacing_args:TF
545         { \exp_not:n }
546         { \_\_hook_double_hashes:n }
547         {#3}
548     }
549     {#1} {#2}
550   }
551 }
```

Adds code to a hook's code pool.

```
552 \cs_new_protected:Npn \_\_hook_prop_gput_labeled_cleanup:nnn #1 #2 #3
553 {
554     \tl_set:Nn \l__hook_return_tl {#1}
555     \_\_hook_if_replacing_args:TF
556     {
557         \_\_hook_if_usable:nT {#2}
558         {
559             \_\_hook_set_normalise_fn:nn {#2}
560             { Invalid~code~added~\msg_line_context: }
561             \_\_hook_normalise_fn:nn {#3} {#1}
562             \prop_get:NnN \l__hook_work_prop {#3} \l__hook_return_tl
563         }
564     }
565     { }
566 \exp_args:NcV \_\_hook_prop_gput_labeled_do:Nnn
567     { g__hook_#2_code_prop } \l__hook_return_tl {#3}
568 }
569 \cs_new_protected:Npn \_\_hook_prop_gput_labeled_do:Nnn #1 #2 #3
570 {
571     \prop_get:NnNTF #1 {#3} \l__hook_return_tl
572     { \prop_gput:Nno #1 {#3} { \l__hook_return_tl #2 } }
573     { \prop_gput:Nnn #1 {#3} {#2} }
574 }
```

`\textrm{(}\textrm{!}\textrm{latextrelease}\textrm{)}\textrm{\EndIncludeInRelease}`

`\textrm{(}\textrm{!}\textrm{latextrelease}\textrm{)}\textrm{\IncludeInRelease}\{2020/10/01\}\textrm{\{}hook_gput_code:nnn}\textrm{\}}`

`\textrm{(}\textrm{!}\textrm{latextrelease}\textrm{)}\textrm{\{}Providing~hooks\}}`

`\textrm{(}\textrm{!}\textrm{latextrelease}\textrm{)}\textrm{\{}cs_gset_protected:Npn \hook_gput_code:nnn #1 #2\}`

`\textrm{(}\textrm{!}\textrm{latextrelease}\textrm{)}\textrm{\{} __hook_normalize_hook_args:Nnn __hook_gput_code:nnn {#1} {#2} \}`

```

580 <latexrelease>\cs_gset_protected:Npn \__hook_gput_code:nnn #1 #2 #3
581 <latexrelease>  {
582 <latexrelease>    \__hook_if_execute_immediately:nTF {#1}
583 <latexrelease>    {#3}
584 <latexrelease>    {
585 <latexrelease>      \__hook_if_usable:nTF {#1}
586 <latexrelease>      {
587 <latexrelease>        \__hook_hook_gput_code_do:nnn {#1} {#2} {#3}
588 <latexrelease>        \__hook_update_hook_code:n {#1}
589 <latexrelease>      }
590 <latexrelease>    }
591 <latexrelease>    \__hook_if_disabled:nTF {#1}
592 <latexrelease>    { \msg_error:nnn { hooks } { hook-disabled } {#1} }
593 <latexrelease>    { \__hook_try_declarngeneric_hook:nnn {#1} {#2} {#3} }
594 <latexrelease>  }
595 <latexrelease> }
596 <latexrelease> }
597 <latexrelease>\cs_gset_protected:Npn \__hook_hook_gput_code_do:nnn #1 #2 #3
598 <latexrelease> {
599 <latexrelease>   \__hook_debug:n{\iow_term:x{****~ Add~ to~}
600 <latexrelease>   \__hook_if_usable:nF {#1} { undeclared~ }
601 <latexrelease>   hook~ #1~ (#2)
602 <latexrelease>   \on@line\space <-- \tl_to_str:n{#3} }
603 <latexrelease> \str_if_eq:nnTF {#2} { top-level }
604 <latexrelease> {
605 <latexrelease>   \str_if_eq:eeTF { top-level } { \__hook_curname_or_default: }
606 <latexrelease>   {
607 <latexrelease>     \__hook_init_structure:n {#1}
608 <latexrelease>     \__hook_tl_gput_right:cn { __hook_toplevel~#1 } {#3}
609 <latexrelease>   }
610 <latexrelease>   { \msg_error:nnn { hooks } { misused-top-level } {#1} }
611 <latexrelease> }
612 <latexrelease> {
613 <latexrelease>   \prop_get:cnNTF { g__hook_#1_code_prop } {#2} \l__hook_return_tl
614 <latexrelease>   {
615 <latexrelease>     \prop_gput:cno { g__hook_#1_code_prop } {#2}
616 <latexrelease>     { \l__hook_return_tl #3 }
617 <latexrelease>   }
618 <latexrelease>   { \prop_gput:cnn { g__hook_#1_code_prop } {#2} {#3} }
619 <latexrelease> }
620 <latexrelease> }
621 <latexrelease>\cs_gset_protected:Npn \hook_gput_code_with_args:nnn #1#2#3 { }
622 <latexrelease>\EndIncludeInRelease

```

(End of definition for \hook_gput_code:nnn and others. These functions are documented on page 204.)

__hook_chk_args_allowed:nn

This macro checks if it is possible to add code with references to a hook's arguments for hook #1. It only does something if the function being run is `replacing_args`. This macro will error if the hook is declared and takes no arguments, then it will set `__hook_replacing_args_false`: so that the macro which called it will add the code normally.

```

623 <latexrelease>\IncludeInRelease{2023/06/01}{\__hook_chk_args_allowed:nn}
624 <latexrelease>          {Hooks~with~args}
625 \cs_new_protected:Npn \__hook_chk_args_allowed:nn #1 #2

```

```

626   {
627     \__hook_if_replacing_args:TF
628     {
629       \__hook_if_declared:nT {#1}
630       { \tl_if_empty:cT { c__hook_#1_parameter_tl } { \use_i:nn } }
631       \use_none:n
632       {
633         \msg_error:nnnn { hooks } { without-args } {#1} {#2}
634         \__hook_replacing_args_false:
635       }
636     }
637   }
638 }
639 \EndIncludeInRelease
640 \IncludeInRelease{2020/10/01}{\__hook_chk_args_allowed:nn}
641 \EndIncludeInRelease
642 \cs_undefine:N \__hook_chk_args_allowed:nn
643 \EndIncludeInRelease

```

(End of definition for `__hook_chk_args_allowed:nn`.)

`__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. An implicitly declared hook doesn't have arguments (in principle), so use `\c_false_bool` here.

```

644 \cs_new_protected:Npn \__hook_gput_undeclared_hook:nnn #1 #2 #3
645   {
646     \__hook_init_structure:n {#1}
647     \__hook_gput_code_do:nnn {#1} {#2} {#3}
648   }

```

(End of definition for `__hook_gput_undeclared_hook:nnn`.)

`__hook_try_declarng_generic_hook:nnn`
`__hook_try_declarng_generic_next_hook:nn`

These entry-level macros just pass the arguments along to the common `__hook_try_declarng_generic_hook:nNNnn` with the right functions to execute when some action is to be taken.

The wrapper `__hook_try_declarng_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_declarng_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.

```

649 \EndIncludeInRelease{2023/06/01}{\__hook_try_declarng_generic_hook:nnn}
650 \EndIncludeInRelease
651 \cs_new_protected:Npn \__hook_try_declarng_generic_hook:nnn #1
652   {
653     \__hook_try_declarng_generic_hook:wnTF #1 / / / \scan_stop: {#1}
654     \__hook_gput_code:nnn
655     \__hook_gput_undeclared_hook:nnn
656     {#1}
657   }
658 \cs_new_protected:Npn \__hook_try_declarng_generic_next_hook:nn #1

```

```

659   {
660     \__hook_try_declarng_generic_hook:wnTF #1 / / / \scan_stop: {#1}
661     \__hook_gput_next_code:nn
662     \__hook_gput_next_do:nn
663     {#1}
664   }
665 \EndIncludeInRelease
666 \IncludeInRelease{2021/11/15}{\__hook_try_declarng_generic_hook:nnn}
667 \Standardise-generic-hook-names
668 \cs_gset_protected:Npn \__hook_try_declarng_generic_hook:nnn #1
669 \{
670   \__hook_try_declarng_generic_hook:wnTF #1 / / / \scan_stop: {#1}
671   \hook_gput_code:nnn
672   \__hook_gput_undeclared_hook:nnn
673   {#1}
674 \}
675 \cs_gset_protected:Npn \__hook_try_declarng_generic_next_hook:nn #1
676 \{
677   \__hook_try_declarng_generic_hook:wnTF #1 / / / \scan_stop: {#1}
678   \hook_gput_code:nn
679   \__hook_gput_next_do:nn
680   {#1}
681 \}
682 \EndIncludeInRelease
683 \IncludeInRelease{2020/10/01}{\__hook_try_declarng_generic_hook:nnn}
684 \Standardise-generic-hook-names
685 \cs_new_protected:Npn \__hook_try_declarng_generic_hook:nnn #1
686 \{
687   \__hook_try_declarng_generic_hook:nNNnn {#1}
688   \hook_gput_code:nnn \__hook_gput_undeclared_hook:nnn
689 \}
690 \cs_new_protected:Npn \__hook_try_declarng_generic_next_hook:nn #1
691 \{
692   \__hook_try_declarng_generic_hook:nNNnn {#1}
693   \hook_gput_code:nn \__hook_gput_next_do:nn
694 \}

```

(End of definition for `__hook_try_declarng_generic_hook:nnn` and
`__hook_try_declarng_generic_next_hook:nn`.)

`__hook_try_declarng_generic_hook:nNNnn` now splits the hook name at the first / (if any) and first checks if it is a file-specific hook (they require some normalization) using `__hook_if_file_hook:wTF`. If not then check it is one of a predefined set for generic names. We also split off the second component to see if we have to make a reversed hook. In either case the function returns `<true>` for a generic hook and `<false>` in other cases.

```

695 \cs_new_protected:Npn \__hook_try_declarng_generic_hook:nNNnn #1
696 \{
697   \__hook_if_file_hook:wTF #1 / \s__hook_mark
698   \{
699     \exp_args:Ne \__hook_try_declarng_generic_hook_split:nNNnn
700     { \exp_args:Ne \__hook_file_hook_normalize:n {#1} }
701   \}
702   { \__hook_try_declarng_generic_hook_split:nNNnn {#1} }
703 \}

```

```

704 <latexrelease>\cs_new_protected:Npn \__hook_try_declaring_generic_hook_split:nNNnn #1 #2 #3
705 <latexrelease>  {
706 <latexrelease>    \__hook_try_declaring_generic_hook:wnTF #1 // \scan_stop: {#1}
707 <latexrelease>    { #2 }
708 <latexrelease>    { #3 } {#1}
709 <latexrelease>  }
710 <latexrelease>\EndIncludeInRelease

(End of definition for \__hook_try_declaring_generic_hook:nNNnn and
\__hook_try_declaring_generic_hook_split:nNNnn.)

```

```

\__hook_try_declaring_generic_hook:wnTF
711 <latexrelease>\IncludeInRelease{2023/06/01}{\__hook_try_declaring_generic_hook:wn}
712 <latexrelease>          {Hooks~with~args}
713 \prg_new_protected_conditional:Npnn \__hook_try_declaring_generic_hook:wn
714   #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
715   {
716     \__hook_if_generic:nTF {#5}
717     {
718       \__hook_if_usable:nF {#5}
719       {

```

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 `ltxcmdhooks`) will generate an appropriate error message.

```

720           \str_if_eq:nnT {#1} { cmd }
721           {
722             \__hook_try_put_cmd_hook:n {#5}
723             \__hook_make_usable:nn {#5} { 9 }
724             \use_none:nnn
725           }

```

Declare the hook always even if it can't really be used (error message generated elsewhere).

Here we use __hook_make_usable:nn, so that a \hook_new:n is still possible later. Generic hooks (except cmd hooks) take no arguments, so use zero as the second argument.

```

726           \__hook_make_usable:nn {#5} { 0 }
727           }
728           \__hook_if_generic_reversed:nT {#5}
729           { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
730           \prg_return_true:
731         }
732         {

```

Generic hooks are all named `<type>/<name>/<place>`, where `<type>` and `<place>` are predefined (`\c__hook_generic_<type>/./<place>_t1`), and `<name>` is the variable component. Older releases had some hooks with the `<name>` in the third part, so the code below supports that syntax for a while, with a warning.

The `\exp_after:wN ... \exp:w` trick is there to remove the conditional structure inserted by __hook_try_declaring_generic_hook:wnTF and thus allow access to the tokens that follow it, as is needed to keep things going.

When the deprecation cycle ends, the lines below should all be replaced by `\prg_return_false:`.

```

733     \__hook_if_deprecated_generic:nTF {#5}
734     {
735         \__hook_DEPRECATED_GENERIC_WARN:n {#5}
736         \exp_after:wN \__hook_DECLARE_DEPRECATED_GENERIC:NNn
737         \exp:w % \exp_end:
738     }
739     { \prg_return_false: }
740 }
741 }
```

`__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).

```

742 \cs_new_protected:Npn \__hook_DEPRECATED_GENERIC_WARN:n #1
743   { \__hook_DEPRECATED_GENERIC_WARN:w #1 \s__hook_mark }
744 \cs_new_protected:Npn \__hook_DEPRECATED_GENERIC_WARN:w
745   #1 / #2 / #3 \s__hook_mark
746   {
747     \ifcsexist:w __hook~#1/#2/#3 \cs_end: \else:
748       \msg_warning:nnnn { hooks } { generic-deprecated } {#1} {#2} {#3}
749     \fi:
750     \cs_gset_eq:cN { __hook~#1/#2/#3 } \scan_stop:
751 }
```

Now that the user has been told about the deprecation, we proceed by swapping `{name}` and `{place}` and adding the code to the correct hook.

```

52 \cs_new_protected:Npn \__hook_DO_DEPRECATED_GENERIC:Nn #1 #2
53   { \__hook_DO_DEPRECATED_GENERIC:Nw #1 #2 \s__hook_mark }
54 \cs_new_protected:Npn \__hook_DO_DEPRECATED_GENERIC:Nw #1
55   #2 / #3 / #4 \s__hook_mark
56   { #1 { #2 / #4 / #3 } }
57 \cs_new_protected:Npn \__hook_DECLARE_DEPRECATED_GENERIC>NNn #1 #2 #3
58   { \__hook_DECLARE_DEPRECATED_GENERIC>NNw #1 #2 #3 \s__hook_mark }
59 \cs_new_protected:Npn \__hook_DECLARE_DEPRECATED_GENERIC>NNw #1 #2
60   #3 / #4 / #5 \s__hook_mark
61   {
62     \__hook_TRY_DECLARING_GENERIC_HOOK:wnTF #3 / #5 / #4 / \scan_stop:
63     { #3 / #5 / #4 }
64     #1 #2 { #3 / #5 / #4 }
65   }
66 \end{InRelease}
```

767 \IncludeInRelease{2021/11/15}{__hook_TRY_DECLARING_GENERIC_HOOK:wn}
768 \Standardise-generic-hook-names
769 \prg_new_protected_conditional:Npnn __hook_TRY_DECLARING_GENERIC_HOOK:wn
770 #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
771 {
772 __hook_if_generic:nTF {#5}
773 {
774 __hook_if_usable:nF {#5}
775 {
776 \str_if_eq:nnT {#1} { cmd }
777 { __hook_TRY_PUT_CMD_HOOK:n {#5} }
778 __hook_make_usable:n {#5}

```

779 <|latexrelease>          }
780 <|latexrelease>          \_\_hook\_if\_generic\_reversed:nT {#5}
781 <|latexrelease>          { \tl_gset:cn { g\_hook\_#5\_reversed_tl } { - } }
782 <|latexrelease>          \prg_return_true:
783 <|latexrelease>        }
784 <|latexrelease>      {
785 <|latexrelease>          \_\_hook_if_deprecated_generic:nTF {#5}
786 <|latexrelease>          {
787 <|latexrelease>              \_\_hook_DEPRECATED_GENERIC_WARN:n {#5}
788 <|latexrelease>              \exp_after:wN \_\_hook_declare_DEPRECATED_GENERIC:Nnn
789 <|latexrelease>              \exp:w % \exp_end:
790 <|latexrelease>          }
791 <|latexrelease>          { \prg_return_false: }
792 <|latexrelease>      }
793 <|latexrelease>  }
794 <|latexrelease> \EndIncludeInRelease

795 <|latexrelease> \IncludeInRelease{2021/06/01}{\_\_hook_try_declaring_generic_hook:wn}
796 <|latexrelease>           {Support-cmd-hooks}
797 <|latexrelease> \prg_new_protected_conditional:Npnn \_\_hook_try_declaring_generic_hook:wn
798 <|latexrelease>     #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
799 <|latexrelease>  {
800 <|latexrelease>      \tl_if_empty:nTF {#2}
801 <|latexrelease>          { \prg_return_false: }
802 <|latexrelease>          {
803 <|latexrelease>              \prop_if_in:NnTF \c__hook_genrics_prop {#1}
804 <|latexrelease>              {
805 <|latexrelease>                  \_\_hook_if_usable:nF {#5}
806 <|latexrelease>                  {
807 <|latexrelease>                      \str_if_eq:nnT {#1} { cmd }
808 <|latexrelease>                      { \_\_hook_try_put_cmd_hook:n {#5} }
809 <|latexrelease>                      \_\_hook_make_usable:n {#5}
810 <|latexrelease>                  }
811 <|latexrelease>              \prop_if_in:NnTF \c__hook_genrics_reversed_ii_prop {#2}
812 <|latexrelease>                  { \tl_gset:cn { g\_hook\_#5\_reversed_tl } { - } }
813 <|latexrelease>                  {
814 <|latexrelease>                      \prop_if_in:NnT \c__hook_genrics_reversed_iii_prop {#3}
815 <|latexrelease>                      { \tl_gset:cn { g\_hook\_#5\_reversed_tl } { - } }
816 <|latexrelease>                  }
817 <|latexrelease>          \prg_return_true:
818 <|latexrelease>      }
819 <|latexrelease>          { \prg_return_false: }
820 <|latexrelease>      }
821 <|latexrelease>  }
822 <|latexrelease> \EndIncludeInRelease

823 <|latexrelease> \IncludeInRelease{2020/10/01}{\_\_hook_try_declaring_generic_hook:wn}
824 <|latexrelease>           {Support-cmd-hooks}
825 <|latexrelease> \prg_new_protected_conditional:Npnn \_\_hook_try_declaring_generic_hook:wn
826 <|latexrelease>     #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
827 <|latexrelease>  {
828 <|latexrelease>      \tl_if_empty:nTF {#2}
829 <|latexrelease>          { \prg_return_false: }
830 <|latexrelease>          {
831 <|latexrelease>              \prop_if_in:NnTF \c__hook_genrics_prop {#1}

```

```

832 〈latexrelease〉      {
833 〈latexrelease〉      \__hook_if_declared:nF {#5} { \hook_new:n {#5} }
834 〈latexrelease〉      \prop_if_in:NnTF \c_hook_generics_reversed_ii_prop {#2}
835 〈latexrelease〉      { \tl_gset:cn { g_hook_#5_reversed_tl } { - } }
836 〈latexrelease〉      {
837 〈latexrelease〉      \prop_if_in:NnT \c_hook_generics_reversed_iii_prop {#3}
838 〈latexrelease〉      { \tl_gset:cn { g_hook_#5_reversed_tl } { - } }
839 〈latexrelease〉      }
840 〈latexrelease〉      \prg_return_true:
841 〈latexrelease〉      }
842 〈latexrelease〉      { \prg_return_false: }
843 〈latexrelease〉      }
844 〈latexrelease〉      }
845 〈latexrelease〉\EndIncludeInRelease

```

(End of definition for `__hook_try_declarng_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>`.

```

846 〈latexrelease〉\IncludeInRelease{2021/11/15}{\__hook_if_file_hook:w}
847 〈latexrelease〉          {Standardise-generic~hook~names}
848 〈latexrelease〉\EndIncludeInRelease
849 〈latexrelease〉\IncludeInRelease{2020/10/01}{\__hook_if_file_hook:w}
850 〈latexrelease〉          {Standardise-generic~hook~names}
851 〈latexrelease〉\prg_new_conditional:Npnn \__hook_if_file_hook:w
852 〈latexrelease〉      #1 / #2 / #3 \s_hook_mark { TF }
853 〈latexrelease〉      {
854 〈latexrelease〉          \str_if_eq:nnTF {#1} { file }
855 〈latexrelease〉          {
856 〈latexrelease〉              \bool_lazy_or:nnTF
857 〈latexrelease〉              { \tl_if_empty_p:n {#3} }
858 〈latexrelease〉              { \str_if_eq_p:nn {#3} { / } }
859 〈latexrelease〉          { \prg_return_false: }
860 〈latexrelease〉          {
861 〈latexrelease〉              \prop_if_in:NnTF \c_hook_generics_file_prop {#2}
862 〈latexrelease〉              { \prg_return_true: }
863 〈latexrelease〉              { \prg_return_false: }
864 〈latexrelease〉          }
865 〈latexrelease〉      }
866 〈latexrelease〉      { \prg_return_false: }
867 〈latexrelease〉      }
868 〈latexrelease〉\EndIncludeInRelease

```

(End of definition for `__hook_if_file_hook:wTF`.)

```

\__hook_file_hook_normalize:n
\__hook_strip_double_slash:n
\__hook_strip_double_slash:w

```

```

869 〈latexrelease〉\IncludeInRelease{2021/11/15}{\__hook_file_hook_normalize:n}
870 〈latexrelease〉          {Standardise-generic~hook~names}
871 〈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}`.

```

872 <latexrelease>\IncludeInRelease{2020/10/01}{\_\_hook_file_hook_normalize:n}
873 <latexrelease>                                {Standardise-generic-hook-names}
874 <latexrelease>\cs_new:Npn \_\_hook_file_hook_normalize:n #1
875 <latexrelease>  { \_\_hook_strip_double_slash:n {#1} }
876 <latexrelease>\cs_new:Npn \_\_hook_strip_double_slash:n #1
877 <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.

```

878 <latexrelease>\cs_new:Npn \_\_hook_strip_double_slash:w #1/#2/#3 // #4 \s__hook_mark
879 <latexrelease>  {
880 <latexrelease>    \tl_if_empty:nTF {#4}
881 <latexrelease>      { #1/#2/#3 }
882 <latexrelease>      { \_\_hook_strip_double_slash:w #1/#2/#3 / #4 \s__hook_mark }
883 <latexrelease>  }
884 <latexrelease>\EndIncludeInRelease

```

(End of definition for `__hook_file_hook_normalize:n`, `__hook_strip_double_slash:n`, and `__hook_strip_double_slash:w`.)

```

\c__hook_generic_cmd./before_tl
\c__hook_generic_cmd./after_tl
\c__hook_generic_env./before_tl
\c__hook_generic_env./after_tl
\c__hook_generic_file./before_tl
\c__hook_generic_file./after_tl
\c__hook_generic_package./before_tl
\c__hook_generic_package./after_tl
\c__hook_generic_class./before_tl
\c__hook_generic_class./after_tl
\c__hook_generic_include./before_tl
\c__hook_generic_include./after_tl
\c__hook_generic_env./begin_tl
\c__hook_generic_env./end_tl
\c__hook_generic_include./end_tl

```

Token lists defining the possible generic hooks. We don't provide any user interface to this as this is meant to be static.

cmd The generic hooks used for commands.

env The generic hooks used in `\begin` and `\end`.

file, package, class, include The generic hooks used when loading a file

```

885 <latexrelease>\IncludeInRelease{2021/11/15}{\c__hook_generic_props}
886 <latexrelease>                                {Standardise-generic-hook-names}
887 \clist_map_inline:nn { cmd , env , file , package , class , include }
888  {
889    \tl_const:cn { c__hook_generic_#/./before_tl } { + }
890    \tl_const:cn { c__hook_generic_#/./after_tl } { - }
891  }
892 \tl_const:cn { c__hook_generic_env./begin_tl } { + }
893 \tl_const:cn { c__hook_generic_env./end_tl } { + }
894 \tl_const:cn { c__hook_generic_include./end_tl } { - }
895 \tl_const:cn { c__hook_generic_include./excluded_tl } { + }

```

Deprecated generic hooks:

```

896 \clist_map_inline:nn { file , package , class , include }
897  {
898    \tl_const:cn { c__hook_deprecated_#/./before_tl } { }
899    \tl_const:cn { c__hook_deprecated_#/./after_tl } { }
900  }
901 \tl_const:cn { c__hook_deprecated_include./end_tl } { }
902 <latexrelease>\EndIncludeInRelease

```

```

903 <latexrelease>\IncludeInRelease{2020/10/01}{\c_hook_generics_prop}
904 <latexrelease>                                {Standardise-generic~hook~names}
905 <latexrelease>\prop_const_from_keyval:Nn \c_hook_generics_prop
906 <latexrelease>      {cmd=,env=,file=,package=,class=,include=}
907 <latexrelease>\EndIncludeInRelease

```

(End of definition for `\c_hook_generic_cmd./before_t1` and others.)

The following generic hooks are supposed to use reverse ordering (the `ii` and `iii` names are kept for the deprecation cycle):

```

908 <latexrelease>\IncludeInRelease{2021/11/15}{\c_hook_generics_reversed_ii_prop}
909 <latexrelease>                                {Standardise-generic~hook~names}
910 <latexrelease>\EndIncludeInRelease
911 <latexrelease>\IncludeInRelease{2020/10/01}{\c_hook_generics_reversed_ii_prop}
912 <latexrelease>                                {Standardise-generic~hook~names}
913 <latexrelease>\prop_const_from_keyval:Nn \c_hook_generics_reversed_ii_prop {after=,end=}
914 <latexrelease>\prop_const_from_keyval:Nn \c_hook_generics_reversed_iii_prop {after=}
915 <latexrelease>\prop_const_from_keyval:Nn \c_hook_generics_file_prop {before=,after=}
916 <latexrelease>\EndIncludeInRelease

```

(End of definition for `\c_hook_generics_reversed_ii_prop`, `\c_hook_generics_reversed_iii_prop`, and `\c_hook_generics_file_prop`.)

Token lists defining the number of arguments for a given type of generic hook.

```

917 <latexrelease>\IncludeInRelease{2023/06/01}{\c_hook_parameter_cmd./before_t1}
918 <latexrelease>                                {Hooks~with~args}

```

`cmd` hooks are declared with 9 arguments because they have a variable number of arguments (depending on the command they are attached to), so we use the maximum here.

```

919 \tl_const:cn { \c_hook_parameter_cmd./before_t1 } { #1#2#3#4#5#6#7#8#9 }
920 \tl_const:cn { \c_hook_parameter_cmd./after_t1 } { #1#2#3#4#5#6#7#8#9 }
921 <latexrelease>\EndIncludeInRelease
922 <latexrelease>\IncludeInRelease{2020/10/01}{\c_hook_parameter_cmd./before_t1}
923 <latexrelease>                                {Hooks~with~args}
924 <latexrelease>\EndIncludeInRelease

```

(End of definition for `\c_hook_parameter_cmd./before_t1` and `\c_hook_parameter_cmd./after_t1`.)

`\hook_gremove_code:nn` With `\hook_gremove_code:nn{<hook>}{{<label>}}` any code for `<hook>` stored under `<label>` is removed.

```

925 <latexrelease>\IncludeInRelease{2023/06/01}{\hook_gremove_code:nn}
926 <latexrelease>                                {Hooks~with~args}
927 \cs_new_protected:Npn \hook_gremove_code:nn #1 #2
928   { \__hook_normalize_hook_args:Nnn \__hook_gremove_code:nn {#1} {#2} }
929 \cs_new_protected:Npn \__hook_gremove_code:nn #1 #2
930   {

```

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).

```

931   \__hook_if_structure_exist:nTF {#1}
932   {

```

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>`.

```

933     \str_if_eq:nnTF {#2} {*}
934     {
935         \prop_gclear:c { g__hook_#1_code_prop }
936         \__hook_toplevel_gset:nn {#1} { }
937         \__hook_next_gset:nn {#1} { }
938     }
939 
```

If the label is top-level then clear the token list, as all code there is under the same label.

```

940         \str_if_eq:nnTF {#2} { top-level }
941         {
942             \__hook_toplevel_gset:nn {#1} { } }
943             \prop_gpop:cnNF { g__hook_#1_code_prop } {#2} \l__hook_return_tl
944             { \msg_warning:nnnn { hooks } { cannot-remove } {#1} {#2} }
945         }
946 
```

Finally update the code, if the hook exists.

```

947     \__hook_if_usable:nT {#1}
948     {
949         \__hook_update_hook_code:n {#1} }
950 
```

If the code pool for this hook doesn't exist, show a warning:

```

950     {
951         \__hook_if_deprecated_generic:nTF {#1}
952         {
953             \__hook_deprecated_generic_warn:n {#1}
954             \__hook_do_deprecated_generic:Nn \__hook_gremove_code:nn {#1} {#2}
955         }
956         { \msg_warning:nnnn { hooks } { cannot-remove } {#1} {#2} }
957     }
958 }
959 <|latexrelease>\EndIncludeInRelease
960 <|latexrelease>\IncludeInRelease{2020/10/01}{\hook_gremove_code:nn}
961 <|latexrelease>                                {Hooks~with~args}
962 <|latexrelease>\cs_new_protected:Npn \__hook_gremove_code:nn #1 #2
963 <|latexrelease>  {
964 <|latexrelease>    \__hook_if_structure_exist:nTF {#1}
965 <|latexrelease>    {
966 <|latexrelease>        \str_if_eq:nnTF {#2} {*}
967 <|latexrelease>        {
968 <|latexrelease>            \prop_gclear:c { g__hook_#1_code_prop }
969 <|latexrelease>            \__hook_tl_gclear:c { __hook_toplevel~#1 }
970 <|latexrelease>            \__hook_tl_gclear:c { __hook_next~#1 }
971 <|latexrelease>        }
972 <|latexrelease>        {
973 <|latexrelease>            \str_if_eq:nnTF {#2} { top-level }
974 <|latexrelease>            { \__hook_tl_gclear:c { __hook_toplevel~#1 } } 
```

```

975  \langle latexrelease\rangle          {
976  \langle latexrelease\rangle          \prop_gpop:cnNF { g__hook_#1_code_prop } {#2} \l__hook_return_tl
977  \langle latexrelease\rangle          { \msg_warning:nnnn { hooks } { cannot-remove } {#1} {#2} }
978  \langle latexrelease\rangle          }
979  \langle latexrelease\rangle          }
980  \langle latexrelease\rangle          \l__hook_if_usable:nT {#1}
981  \langle latexrelease\rangle          { \l__hook_update_hook_code:n {#1} }
982  \langle latexrelease\rangle        }
983  \langle latexrelease\rangle        {
984  \langle latexrelease\rangle          \l__hook_if_deprecated_generic:nTF {#1}
985  \langle latexrelease\rangle          {
986  \langle latexrelease\rangle            \l__hook_DEPRECATED_GENERIC_WARN:n {#1}
987  \langle latexrelease\rangle            \l__hook_do_DEPRECATED_GENERIC:Nn \l__hook_gremove_code:nn {#1} {#2}
988  \langle latexrelease\rangle          }
989  \langle latexrelease\rangle          { \msg_warning:nnnn { hooks } { cannot-remove } {#1} {#2} }
990  \langle latexrelease\rangle        }
991  \langle latexrelease\rangle      }
992  \langle latexrelease\rangle \EndIncludeInRelease

```

(End of definition for `\hook_gremove_code:nn` and `\l__hook_gremove_code:nn`. This function is documented on page [205](#).)

```

\l__hook_cs_gput_right:nnn
  \l__hook_cs_gput_right_fast:nnn
  \l__hook_cs_gput_right_slow:nnn
\l__hook_code_gset_auxi:nnnn
\l__hook_code_gset_auxi:eeen

```

This macro is used to append code to the `toplevel` and `next` token lists, trating them correctly depending on their number of arguments, and depending if the code being added should have parameter tokens understood as parameters, or doubled to be stored as parameter tokens.

```

993  \langle latexrelease\rangle \IncludeInRelease{2023/06/01}{\l__hook_cs_gput_right:nnn}
994  \langle latexrelease\rangle           {Hooks~with~args}

```

Check if the current hook is declared and takes no arguments. In this case, we short-circuit and use the simpler and much faster approach that doesn't require hash-doubling.

```

995  \cs_new_protected:Npn \l__hook_cs_gput_right:nnn #1 #2
996  {
997    \if:w T
998      \l__hook_if_declared:nF {#2} { F }
999      \tl_if_empty:cF { c__hook_#2_parameter_tl } { F }
1000     T
1001     \exp_after:wN \l__hook_cs_gput_right_fast:nnn
1002   \else:
1003     \exp_after:wN \l__hook_cs_gput_right_slow:nnn
1004   \fi:
1005     {#1} {#2}
1006   }
1007 \cs_new_protected:Npn \l__hook_cs_gput_right_fast:nnn #1 #2 #3
1008   { \cs_gset:cp { _hook#1~#2 } { \exp_not:v { _hook#1~#2 } \exp_not:n {#3} } }
1009 \cs_new_protected:Npn \l__hook_cs_gput_right_slow:nnn #1 #2 #3
1010   {

```

The auxiliary `\l__hook_code_gset_auxi:eeen` just does the assignment at the end. Its first argument is the parameter text of the macro, which is chosen here depending if `\c_-_hook_{hook}_parameter_tl` exists, if the hook is declared, and if it's a generic hook.

```

1011   \cs_if_exist:cF { _hook#1~#2 }
1012     { \l__hook_code_gset_aux:nnn {#1} {#2} { } }
1013   \l__hook_code_gset_auxi:eeen
1014   {

```

```

1015     \__hook_if_declared:nTF {#2}
1016     { \tl_use:c { c__hook_#2_parameter_tl } }
1017     {
1018         \__hook_if_generic:nTF {#2}
1019         { \__hook_generic_parameter:n {#2} }
1020         { \c__hook_nine_parameters_tl }
1021     }
1022 }
```

Here we take the existing code in the macro, expand it with as many arguments as it takes, then double the hashes so the code can be reused.

PhO: Maybe can be improved.
The case of adding to an empty cs
can be optimised by quickly checking
ing \cs_replacement_spec.

```

1023     {
1024         \exp_args:NNo \exp_args:No \__hook_double_hashes:n
1025         {
1026             \cs:w __hook#1~#2 \exp_last_unbraced:Ne \cs_end:
1027             { \__hook_braced_cs_parameter:n { __hook#1~#2 } }
1028         }
1029     }
```

Now the new code: if we are replacing arguments, then hashes are left untouched, otherwise they are doubled.

```

1030     {
1031         \__hook_if_replacing_args:TF
1032         { \exp_not:n }
1033         { \__hook_double_hashes:n }
1034         {#3}
1035     }
```

And finally, the csname which we'll define with all the above.

```

1036     { __hook#1~#2 }
1037 }
```

And as promised, the auxiliary that does the definition.

```

1038 \cs_new_protected:Npn \__hook_code_gset_auxi:nnnn #1 #2 #3 #4
1039   { \cs_gset:cpn {#4} #1 { #2 #3 } }
1040 \cs_generate_variant:Nn \__hook_code_gset_auxi:nnnn { een }
1041 \end{macro}
1042 \end{macro}
1043 \end{macro}
1044 \end{macro}
1045 \end{macro}
1046 \end{macro}
1047 \end{macro}
1048 \end{macro}
```

(End of definition for __hook_code_gset_auxi:nnnn and others.)

These macros define __hook<type>_<hook> (with <type> being _next, _toplevel, or empty) with the given code and the parameters stored in \c__hook_<hook>_parameter_tl (or none, if that doesn't exist).

```

1049 \end{macro}
1050 \end{macro}
1051 \end{macro}
1052 \end{macro}
1053 \end{macro}
```

```

1054 { \__hook_code_gset_aux:nnn { _toplevel } }
1055 \cs_new_protected:Npn \__hook_next_gset:nn
1056 { \__hook_code_gset_aux:nnn { _next } }
1057 \cs_new_protected:Npn \__hook_code_gset_aux:nnn #1 #2 #3
1058 {
1059     \cs_gset:cpn { __hook#1~#2 \exp_last_unbraced:Ne }
1060     { \__hook_parameter:n {#2} }
1061     {#3}
1062 }
1063 \cs_generate_variant:Nn \__hook_code_gset:nn { ne }
1064 ⟨latexrelease⟩\EndIncludeInRelease
1065 ⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_code_gset:nn}
1066 ⟨latexrelease⟩
1067 ⟨latexrelease⟩\cs_undefine:N \__hook_code_gset:nn
1068 ⟨latexrelease⟩\cs_undefine:N \__hook_toplevel_gset:nn
1069 ⟨latexrelease⟩\cs_undefine:N \__hook_next_gset:nn
1070 ⟨latexrelease⟩\cs_undefine:N \__hook_code_gset_aux:nnn
1071 ⟨latexrelease⟩\EndIncludeInRelease

```

(End of definition for `__hook_code_gset:nn` and others.)

`__hook_normalise_cs_args:nn`

This macro normalises the parameters of the macros `__hook<type>_<hook>` to take the right number of arguments after a hook is declared. At this point we know `\c__-hook_<hook>_parameter_tl` exists, so use that to count the arguments and use that as `<parameter text>` for the newly (re)defined macro.

```

1072 ⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_normalise_cs_args:nn}
1073 ⟨latexrelease⟩
1074 \cs_new_protected:Npn \__hook_normalise_cs_args:nn #1 #2
1075 {
1076     \cs_if_exist:cT { __hook#1~#2 }
1077     {
1078         \__hook_code_gset_auxi:een
1079         { \tl_use:c { c__hook_#2_parameter_tl } }
1080         {
1081             \exp_args:NNo \exp_args:No \__hook_double_hashes:n
1082             {
1083                 \cs:w __hook#1~#2 \exp_last_unbraced:Ne \cs_end:
1084                 { \__hook_braced_cs_parameter:n { __hook#1~#2 } }
1085             }
1086         }
1087         { }
1088         { __hook#1~#2 }
1089     }
1090 }
1091 ⟨latexrelease⟩\EndIncludeInRelease
1092 ⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_normalise_cs_args:nn}
1093 ⟨latexrelease⟩
1094 ⟨latexrelease⟩\cs_undefine:N \__hook_normalise_cs_args:nn
1095 ⟨latexrelease⟩\EndIncludeInRelease

```

(End of definition for `__hook_normalise_cs_args:nn`.)

`__hook_normalise_code_pool:n`
`__hook_set_normalise_fn:nn`

This one's a bit of a hack. It takes a hook, and iterates over its code pool (`\g__hook-<hook>_code_prop`), redefining each code label to use only valid arguments. This is used

when, for example, a code is added referencing arguments #1 and #2, but the hook has only #1. In this example, every reference to #2 is changed to ##2. This is done because otherwise TeX will throw a low-level error every time some change happens to the hook (code is added, a rule is set, etc), which can get quite repetitive for no good reason.

```
1096 <latexrelease>\IncludeInRelease{2023/06/01}{\_\_hook_normalise_code_pool:n}
1097 <latexrelease>                                {Hooks~with~args}
1098 \cs_new_protected:Npn \_\_hook_normalise_code_pool:n #1
1099 {
```

First, call __hook_set_normalise_fn:nn with the hook name to set everything up, then we'll loop over the hook's code pool applying the normalisation above. After that's done, copy the temporary property list back to the hook's.

```
1100   \_\_hook_set_normalise_fn:nn {#1} { Offending-label:~'##1' }
1101   \prop_clear:N \l_\_hook_work_prop
1102   \prop_map_function:cN { g_\_hook_#1_code_prop } \_\_hook_normalise_fn:nn
1103   \prop_gset_eq:cN { g_\_hook_#1_code_prop } \l_\_hook_work_prop
1104 }
```

The sole purpose of this function is to define __hook_normalise_fn:nn, which will then do the correcting of the code being added to the hook.

```
1105 \cs_new_protected:Npn \_\_hook_set_normalise_fn:nn #1 #2
1106 {
```

To start, we define two auxiliary token lists. \l__hook_tmpb_tl contains:

```
{\c_\_hook_hashes_tl 1}
{\c_\_hook_hashes_tl 2}
...
{\c_\_hook_hashes_tl 9}

1107 \cs_set:Npn \_\_hook_tmp:w ##1##2##3##4##5##6##7##8##9 { }
1108 \tl_set:Ne \l_\_hook_tmpb_tl
1109 { \_\_hook_braced_cs_parameter:n { \_\_hook_tmp:w } }
1110 \group_begin:
1111   \_\_hook_tl_set:cn { c_\_hook_hash_tl } { \exp_not:N \c_\_hook_hashes_tl }
1112   \use:e
1113   {
1114 \group_end:
1115 \tl_set:Nn \exp_not:N \l_\_hook_tmpb_tl { \l_\_hook_tmpb_tl }
1116 }
```

And \l__hook_tmpa_tl contains:

```
{\c_\_hook_hash_tl 1}
{\c_\_hook_hash_tl 2}
...
{\c_\_hook_hash_tl <n>}
```

with *<n>* being the number of arguments declared for the hook.

```
1117 \exp_last_unbraced:Nnf
1118 \cs_set:Npn \_\_hook_tmp:w { \_\_hook_parameter:n {#1} } { }
1119 \tl_set:Ne \l_\_hook_tmpa_tl { \_\_hook_braced_cs_parameter:n { \_\_hook_tmp:w } }
```

Now this function does the fun part. It is meant to be used with \prop_map_function:NN, taking a label name in ##1 and the code stored in that label in ##2.

```
1120 \cs_gset_protected:Npx \_\_hook_normalise_fn:nn ##1 ##2
1121 {
```

Here we'll define two auxiliary macros: the first one throws an error when it detects an invalid argument reference. It piggybacks on TeX's low-level "Illegal parameter number" error, but it defines a weirdly-named control sequence so that the error comes out nicely formatted. For example, if the label "badpkg" adds some code that references argument #3 in the hook "foo", which takes only two arguments, the error will be:

```
! Illegal parameter number in definition of hook 'foo'.
(hooks)          Offending label: 'badpkg'.
<to be read again>
            3
```

At the point of this definition, the error is raised if the code happens to reference an invalid argument. If it was possible to detect that this definition raised no error, the next step would be unnecessary. We'll do all this in a group so this weird definition doesn't leak out, and set `\tex_escapechar:D` to `-1` so this hack shows up extra nice in the case of an error.

```
1122      \group_begin:
1123          \int_set:Nn \tex_escapechar:D { -1 }
1124          \cs_set:cpn
1125              {
1126                  hook~'#1'. ^~J
1127                  (hooks) \prg_replicate:nn { 13 } { ~ }
1128                  #2 % more message text
1129              }
1130          \exp_not:v { c__hook_#1_parameter_tl }
1131 {##2}
1132      \group_end:
```

This next macro, with a much less fabulous name, takes always nine arguments, and it just transfers the code `##2` under the label `##1` to the temporary property list. The first $\langle n \rangle$ arguments are taken from `\l__hook_tmpa_tl`, and the other $9 - \langle n \rangle$ taken from `\l__hook_tmpb_tl` (which contains twice as many # tokens as the former). Then, `__hook_double_hashes:n` is used to double non-argument hashes, and expand the `\c__hook_hash_tl` and `\c__hook_hashes_tl` to the actual parameter tokens.

```
1133      \cs_set:Npn \exp_not:N \__hook_tmp:w
1134          \exp_not:V \c__hook_nine_parameters_tl
1135      {
1136          \prop_put:Nne \exp_not:N \l__hook_work_prop
1137          {##1} { \exp_not:N \__hook_double_hashes:n {##2} }
1138      }
```

This next macro, with a much less fabulous name, takes always nine arguments, and it just transfers the code `##2` under the label `##1` to the temporary property list. The first $\langle n \rangle$ arguments are taken from `\l__hook_tmpa_tl`, and the other $9 - \langle n \rangle$ taken from `\l__hook_tmpb_tl` (which contains twice as many # tokens as the former). Then, `__hook_double_hashes:n` is used to double non-argument hashes, and expand the `\c__hook_hash_tl` and `\c__hook_hashes_tl` to the actual parameter tokens.

```
1139      \exp_not:N \__hook_tmp:w
1140          \exp_not:V \l__hook_tmpa_tl
1141          \exp_args:No \exp_not:o
1142          { \exp_after:wN \__hook_tmp:w \l__hook_tmpb_tl }
1143      }
1144  }
```

```

1145 \cs_new_eq:NN \__hook_normalise_fn:nn ?
1146 ⟨latexrelease⟩\EndIncludeInRelease
1147 ⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\__hook_normalise_code_pool:n}
1148 ⟨latexrelease⟩
1149 ⟨latexrelease⟩\cs_undefine:N \__hook_normalise_code_pool:n
1150 ⟨latexrelease⟩\EndIncludeInRelease

```

Check if the expansion of a control sequence is empty by looking at its replacement text.

```

\__hook_cs_if_empty_p:c
\__hook_cs_if_empty:cTF
1151 ⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_cs_if_empty:c}
1152 ⟨latexrelease⟩
1153 {Hooks~with~args}
1154 \prg_new_conditional:Npnn \__hook_cs_if_empty:c #1 { p, T, F, TF }
1155 {
1156     \if:w \scan_stop: \__hook_replacement_spec:c {#1} \scan_stop:
1157         \prg_return_true:
1158     \else:
1159         \prg_return_false:
1160     \fi:
1161 }
1162 \cs_new:Npn \__hook_replacement_spec:c #1
1163 {
1164     \exp_args:Nc \token_if_macro:NT {#1}
1165     { \cs_replacement_spec:c {#1} }
1166 }
1167 ⟨latexrelease⟩\EndIncludeInRelease
1168 ⟨latexrelease⟩
1169 ⟨latexrelease⟩\cs_undefine:N \__hook_cs_if_empty:c
1170 ⟨latexrelease⟩\EndIncludeInRelease

```

(End of definition for `__hook_normalise_code_pool:n`, `__hook_set_normalise_fn:nn`, and
`__hook_cs_if_empty:cTF`.)

Looks at the *parameter text* of a control sequence, and returns a run of “hidden” braced parameters for that macro. This works as long as the macros take a simple run of zero to nine arguments. The parameters are “hidden” because the parameter tokens are returned inside `\c__hook_hash_tl` instead of explicitly, so that `__hook_double_hashes:n` won’t touch these.

```

\__hook_braced_cs_parameter:n
\__hook_braced_hidden_loop:w
\__hook_cs_parameter_count:N
\__hook_cs_parameter_count:w
\__hook_cs_end:w
1171 ⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\__hook_braced_cs_parameter:n}
1172 ⟨latexrelease⟩
1173 {Hooks~with~args}
1174 \cs_new:Npn \__hook_braced_cs_parameter:n #1
1175 {
1176     \exp_last_unbraced:Ne \__hook_braced_hidden_loop:w
1177     { \exp_args:Nc \__hook_cs_parameter_count:N {#1} } ? \s__hook_mark
1178 }
1179 \cs_new:Npn \__hook_braced_hidden_loop:w #1
1180 {
1181     \if:w ? #1
1182         \__hook_use_i_delimit_by_s_mark:nw
1183     \fi:
1184     { \exp_not:N \c__hook_hash_tl #1 }
1185     \__hook_braced_hidden_loop:w
1186 }
1187 
```

```

1186 \cs_new:Npn \__hook_cs_parameter_count:N #1
1187   {
1188     \exp_last_unbraced:Nf \__hook_cs_parameter_count:w
1189     { \token_if_macro:NT #1 { \cs_parameter_spec:N #1 } }
1190     ? \__hook_cs_end:w ? \__hook_cs_end:w ? \__hook_cs_end:w
1191     ? \__hook_cs_end:w ? \__hook_cs_end:w ? \__hook_cs_end:w
1192     ? \__hook_cs_end:w ? \__hook_cs_end:w ? \__hook_cs_end:w
1193     \s__hook_mark
1194   }
1195 \cs_new:Npn \__hook_cs_parameter_count:w #1#2 #3#4 #5#6 #7#8
1196   { #2 #4 #6 #8 \__hook_cs_parameter_count:w }
1197 \cs_new:Npn \__hook_cs_end:w #1 \s__hook_mark { }
1198 \end{IncludeInRelease}

```

This function can't be undefined when rolling back because it's used at the end of this module to adequate the hook data structures to previous versions.

```

1199 \end{IncludeInRelease}{2020/10/01}{\__hook_braced_cs_parameter:n}
1200 \end{IncludeInRelease} {Hooks~with~args}
1201 \end{IncludeInRelease}

```

(End of definition for `__hook_braced_cs_parameter:n` and others.)

`__hook_braced_parameter:n`
`__hook_braced_real_loop:w`

This one is used in simpler cases, where no special handling of hashes is required. This is used only inside `__hook_initialize_hook_code:n`, so it assumes `\c__hook_<hook>_parameter_tl` is defined, but should work otherwise.

```

1202 \end{IncludeInRelease}{2023/06/01}{\__hook_braced_parameter:n}
1203 \end{IncludeInRelease} {Hooks~with~args}
1204 \cs_new:Npn \__hook_braced_parameter:n #1
1205   {
1206     \if_case:w
1207       \int_eval:n
1208         { \exp_args:Nv \str_count:n { \c__hook_#1_parameter_tl } / 3 }
1209       \exp_stop_f:
1210       \or: {##1}
1211       \or: {##1} {##2}
1212       \or: {##1} {##2} {##3}
1213       \or: {##1} {##2} {##3} {##4}
1214       \or: {##1} {##2} {##3} {##4} {##5}
1215       \or: {##1} {##2} {##3} {##4} {##5} {##6}
1216       \or: {##1} {##2} {##3} {##4} {##5} {##6} {##7}
1217       \or: {##1} {##2} {##3} {##4} {##5} {##6} {##7} {##8}
1218       \or: {##1} {##2} {##3} {##4} {##5} {##6} {##7} {##8} {##9}
1219       \else:
1220         \msg_expandable_error:nnn { latex2e } { should-not-happen }
1221           { Invalid~parameter~spec. }
1222         \fi:
1223   }
1224 \end{IncludeInRelease}
1225 \end{IncludeInRelease}{2020/10/01}{\__hook_braced_parameter:n}
1226 \end{IncludeInRelease} {Hooks~with~args}
1227 \end{IncludeInRelease}\cs_undefine:N \__hook_braced_parameter:n
1228 \end{IncludeInRelease}

```

(End of definition for `__hook_braced_parameter:n` and `__hook_braced_real_loop:w`.)

```

\_\_hook\_parameter:n This is just a shortcut to e- or f-expand to the <parameter text> of the hook.
1229 〈latexrelease〉\IncludeInRelease{2023/06/01}{\_\_hook\_parameter:n}
1230 〈latexrelease〉                                {Hooks~with~args}
1231 \cs_new:Npn \_\_hook_parameter:n #1
1232 {
1233     \cs:w c_\_\_hook_
1234     \tl_if_exist:cTF { c_\_\_hook_\#1_parameter_tl }
1235     { #1_parameter } { empty }
1236     _tl \cs_end:
1237 }
1238 \cs_new:Npn \_\_hook_generic_parameter:n #1
1239 { \_\_hook_generic_parameter:w #1 / / \s_\_\_hook_mark }
1240 \cs_new:Npn \_\_hook_generic_parameter:w #1 / #2 / #3 / #4 \s_\_\_hook_mark
1241 {
1242     \cs_if_exist_use:cF { c_\_\_hook_parameter_\#1./.#3_tl }
1243     { \c_\_\_hook_empty_tl }
1244 }
1245 〈latexrelease〉\EndIncludeInRelease
1246 〈latexrelease〉\IncludeInRelease{2020/10/01}{\_\_hook_parameter:n}
1247 〈latexrelease〉                                {Hooks~with~args}
1248 〈latexrelease〉\cs_undefine:N \_\_hook_parameter:n
1249 〈latexrelease〉\cs_undefine:N \_\_hook_generic_parameter:n
1250 〈latexrelease〉\EndIncludeInRelease

(End of definition for \_\_hook_parameter:n.)

```

4.7 Setting rules for hooks code

```

\g_\_\_hook_\?\?_code_prop
  \_\_hook_\?\?
\g_\_\_hook_\?\?_reversed_tl
\c_\_\_hook_\?\?_parameter_tl

```

Initially these variables simply used an empty “label” name (not two question marks). This was a bit unfortunate, because then `13doc` 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.

```

1251 \prop_new:c { g_\_\_hook_\?\?_code_prop }
1252 \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.

```

1253 \tl_new:c { g_\_\_hook_\?\?_reversed_tl }

```

The parameter text for the “default” hook is empty.

```

1254 〈latexrelease〉\IncludeInRelease{2023/06/01}{\c_\_\_hook_\?\?_parameter_tl}
1255 〈latexrelease〉                                {Hooks~with~args}
1256 \tl_const:cn { c_\_\_hook_\?\?_parameter_tl } { }
1257 〈latexrelease〉\EndIncludeInRelease
1258 〈latexrelease〉\IncludeInRelease{2020/10/01}{\c_\_\_hook_\?\?_parameter_tl}
1259 〈latexrelease〉                                {Hooks~with~args}
1260 〈latexrelease〉\cs_undefine:c { c_\_\_hook_\?\?_parameter_tl }
1261 〈latexrelease〉\EndIncludeInRelease

```

(End of definition for \g__hook_??_code_prop and others.)

\hook_gset_rule:nnnn
_hook_gset_rule:nnnn

With \hook_gset_rule:nnnn{\langle hook\rangle}{\langle label1\rangle}{\langle relation\rangle}{\langle label2\rangle} a relation is defined between the two code labels for the given \langle hook\rangle. The special hook ?? stands for any hook, which sets a default rule (to be used if no other relation between the two hooks exist).

```
1262 \cs_new_protected:Npn \hook_gset_rule:nnnn #1#2#3#4
1263 {
1264     \_hook_normalize_hook_rule_args:Nnnnn \_hook_gset_rule:nnnn
1265     {#1} {#2} {#3} {#4}
1266 }
1267 \IfFileExists{2022/06/01}{\_hook_gset_rule:nnnn}
1268 {\RefuseSettingRuleForOneTimeHooks}
1269 \cs_new_protected:Npn \_hook_gset_rule:nnnn #1#2#3#4
1270 {
1271     \_hook_if_deprecated_generic:nT {#1}
1272     {
1273         \_hook_DEPRECATED_WARN:n {#1}
1274         \_hook_do_DEPRECATED_GENERIC:Nn \_hook_gset_rule:nnnn {#1}
1275         {#2} {#3} {#4}
1276         \_hook_use_none_delimit_by_s_mark:w
1277     }
1278     \_hook_if_execute_immediately:nT {#1}
1279     {
1280         \msg_error:nnnnn { hooks } { rule-too-late }
1281         {#1} {#2} {#3} {#4}
1282         \_hook_use_none_delimit_by_s_mark:w
1283     }
1284 }
```

First we ensure the basic data structure of the hook exists:

```
1284     \_hook_init_structure:n {#1}
```

Then we clear any previous relationship between both labels.

```
1285     \_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.

```
1286 \cs_if_exist_use:cTF { \_hook_rule_#3_gset:nnn }
1287 {
1288     {#1} {#2} {#4}
1289     \_hook_update_hook_code:n {#1}
1290 }
1291 {
1292     \msg_error:nnnnn { hooks } { unknown-rule }
1293     {#1} {#2} {#3} {#4}
1294 }
1295 \s__hook_mark
1296 }

1297 \EndIncludeInRelease
1298 \IncludeInRelease{2020/10/01}{\_hook_gset_rule:nnnn}
1299 {\RefuseSettingRuleForOneTimeHooks}
1300 \cs_new_protected:Npn \_hook_gset_rule:nnnn #1#2#3#4
1301 {
1302     \_hook_if_deprecated_generic:nT {#1}
```

```

1303 <|latexrelease>      {
1304   <|latexrelease>      \__hook_deprecated_generic_warn:n {#1}
1305   <|latexrelease>      \__hook_do_DEPRECATED_GENERIC:Nn \__hook_gset_rule:nnnn {#1}
1306   <|latexrelease>      {#2} {#3} {#4}
1307   <|latexrelease>      \exp_after:wN \use:none:nnnnnnnn \use:none:n
1308   <|latexrelease>      }
1309   <|latexrelease>      \__hook_init_structure:n {#1}
1310   <|latexrelease>      \__hook_rule_gclear:nnn {#1} {#2} {#4}
1311   <|latexrelease>      \cs_if_exist_use:cTF { __hook_rule_#3_gset:nnn }
1312   <|latexrelease>      {
1313     <|latexrelease>      {#1} {#2} {#4}
1314     <|latexrelease>      \__hook_update_hook_code:n {#1}
1315   <|latexrelease>      }
1316   <|latexrelease>      {
1317     <|latexrelease>      \msg_error:nnnnn { hooks } { unknown-rule }
1318     <|latexrelease>      {#1} {#2} {#3} {#4}
1319   <|latexrelease>      }
1320   <|latexrelease>  }
1321 <|latexrelease> \EndIncludeInRelease

```

(End of definition for \hook_gset_rule:nnnn and __hook_gset_rule:nnnn. This function is documented on page 205.)

__hook_rule_before_gset:nnn
__hook_rule_after_gset:nnn
__hook_rule_<_gset:nnn
__hook_rule_>_gset:nnn

Then we add the new rule. We need to normalize the rules here to allow for faster processing later. Given a pair of labels l_A and l_B , the rule $l_A > l_B$ is the same as $l_B < l_A$ only presented differently. But by normalizing the forms of the rule to a single representation, say, $l_B < l_A$, reduces the time spent looking for the rules later considerably.

Here we do that normalization by using \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 {\langle hook\rangle} {\langle l_A\rangle} {\langle l_B\rangle} 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.

```

1322 \cs_new_protected:Npn \__hook_rule_before_gset:nnn #1#2#3
1323   {
1324     \__hook_tl_gset:cx { g__hook_#1_rule_ \__hook_label_pair:nn {#2} {#3} _tl }
1325     { \__hook_label_ordered:nnTF {#2} {#3} { < } { > } }
1326   }
1327 \cs_new_eq:cN { __hook_rule_<_gset:nnn } \__hook_rule_before_gset:nnn
1328 \cs_new_protected:Npn \__hook_rule_after_gset:nnn #1#2#3
1329   {
1330     \__hook_tl_gset:cx { g__hook_#1_rule_ \__hook_label_pair:nn {#3} {#2} _tl }
1331     { \__hook_label_ordered:nnTF {#3} {#2} { < } { > } }
1332   }
1333 \cs_new_eq:cN { __hook_rule_>_gset:nnn } \__hook_rule_after_gset:nnn

```

(End of 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.

```

1334 \cs_new_protected:Npn \__hook_rule_voids_gset:nnn #1#2#3
1335   {
1336     \__hook_tl_gset:cx { g__hook_#1_rule_ \__hook_label_pair:nn {#2} {#3} _tl }
1337     { \__hook_label_ordered:nnTF {#2} {#3} { -> } { <- } }
1338   }

```

(End of 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.

```
1339 \cs_new_protected:cpn { __hook_rule_incompatible_error_gset:nnn } #1#2#3
1340   { __hook_tl_gset:cn { g__hook_#1_rule_ __hook_label_pair:nn {#2} {#3} _tl }
1341     { xE } }
1342 \cs_new_protected:cpn { __hook_rule_incompatible_warning_gset:nnn } #1#2#3
1343   { __hook_tl_gset:cn { g__hook_#1_rule_ __hook_label_pair:nn {#2} {#3} _tl }
1344     { xW } }
```

(End of 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.

```
1345 \cs_new_protected:Npn __hook_rule_unrelated_gset:nnn #1#2#3 { }
1346 \cs_new_protected:Npn __hook_rule_gclear:nnn #1#2#3
1347   { \cs_undefine:c { g__hook_#1_rule_ __hook_label_pair:nn {#2} {#3} _tl } }
```

(End of definition for `__hook_rule_unrelated_gset:nnn` and `__hook_rule_gclear:nnn`.)

`__hook_label_pair:nn`

Ensure that the lexically greater label comes first.

```
1348 \cs_new:Npn __hook_label_pair:nn #1#2
1349   {
1350     \if_case:w __hook_str_compare:nn {#1} {#2} \exp_stop_f:
1351       #1 | #1 % 0
1352     \or: #1 | #2 % +1
1353     \else: #2 | #1 % -1
1354     \fi:
1355   }
```

(End of 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.

```
1356 \prg_new_conditional:Npnn __hook_label_ordered:nn #1#2 { TF }
1357   {
1358     \if_int_compare:w __hook_str_compare:nn {#1} {#2} > 0 \exp_stop_f:
1359       \prg_return_true:
1360     \else:
1361       \prg_return_false:
1362     \fi:
1363   }
```

(End of definition for `__hook_label_ordered:nnTF`.)

`__hook_if_label_case:nnnn`

To avoid doing the string comparison twice in `__hook_initialize_single:NNn` (once with `\str_if_eq:nn` and again with `__hook_label_ordered:nn`), we use a three-way branching macro that will compare #1 and #2 and expand to `\use_i:nn` if they are equal, `\use_ii:nn` if #1 is lexically greater, and `\use_iii:nn` otherwise.

```
1364 \cs_new:Npn __hook_if_label_case:nnnn #1#2
1365   {
1366     \cs:w use_
1367       \if_case:w __hook_str_compare:nn {#1} {#2}
```

```

1368         i \or: ii \else: iii \fi: :nnn
1369         \cs_end:
1370     }

```

(End of definition for `_hook_if_label_case:nnnn.`)

`_hook_update_hook_code:n` Before `\begin{document}` this does nothing, in the body it reinitializes the hook code using the altered data.

```

1371 \cs_new_eq:NN \_hook_update_hook_code:n \use_none:n

```

(End of 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.

```

1372 \IncludeInRelease{2023/06/01}{\_hook_initialize_all:}
1373 {Hooks-with-args}
1374 \cs_new_protected:Npn \_hook_initialize_all:
1375 {

```

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.

```

1376 \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. Here we have to determine if the hook has arguments so that auxiliaries know what to do with hashes. We look at `\c_{hook}_parameter_tl`, if it has any parameters, and set `replacing_args` accordingly.

```

1377 \_hook_debug:n { \prop_gclear:N \g__hook_used_prop }
1378 \seq_map_inline:Nn \g__hook_all_seq
1379 {
1380     \tl_if_empty:cTF { c__hook_##1_parameter_tl }
1381     { \_hook_replacing_args_false: }
1382     { \_hook_replacing_args_true: }
1383     \_hook_update_hook_code:n {##1}
1384     \_hook_replacing_args_reset:
1385 }

```

If we are debugging we show results hook by hook for all hooks that have data.

```

1386 \_hook_debug:n
1387 {
1388     \iow_term:x { ^~J All~initialized~(non-empty)~hooks: }
1389     \prop_map_inline:Nn \g__hook_used_prop
1390     {
1391         \iow_term:x
1392         { ^~J ~ ##1 ~ -> ~ \cs_replacement_spec:c { __hook~##1 } ~ }
1393     }
1394 }

```

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).

```

1395 \_hook_post_initialization_defs:
1396 }

```

```

1397  ⟨latexrelease⟩ \EndIncludeInRelease
1398  ⟨latexrelease⟩ \IncludeInRelease{2020/10/01}{\__hook_initialize_all:}
1399  ⟨latexrelease⟩           {Hooks~with~args}
1400  ⟨latexrelease⟩ \cs_gset_protected:Npn \__hook_initialize_all:
1401  ⟨latexrelease⟩   {
1402  ⟨latexrelease⟩     \cs_gset_eq:NN \__hook_update_hook_code:n \__hook_initialize_hook_code:n
1403  ⟨latexrelease⟩     \__hook_debug:n { \prop_gclear:N \g__hook_used_prop }
1404  ⟨latexrelease⟩     \seq_map_inline:Nn \g__hook_all_seq
1405  ⟨latexrelease⟩       { \__hook_update_hook_code:n {##1} }
1406  ⟨latexrelease⟩     \__hook_debug:n
1407  ⟨latexrelease⟩   {
1408  ⟨latexrelease⟩     \iow_term:x{^^JAll~ initialized~ (non-empty)~ hooks:}
1409  ⟨latexrelease⟩     \prop_map_inline:Nn \g__hook_used_prop
1410  ⟨latexrelease⟩       {
1411  ⟨latexrelease⟩         \iow_term:x
1412  ⟨latexrelease⟩           { ^^J ~ ##1 ~ -> ~ \cs_replacement_spec:c { __hook##1 } ~ }
1413  ⟨latexrelease⟩       }
1414  ⟨latexrelease⟩   }
1415  ⟨latexrelease⟩     \cs_gset_eq:NN \hook_use:n \__hook_use_initialized:n
1416  ⟨latexrelease⟩     \cs_gset_eq:NN \__hook_preamble_hook:n \use_none:n
1417  ⟨latexrelease⟩   }
1418  ⟨@=⟩
1419  ⟨latexrelease⟩ \cs_gset_eq:NN \expl@@@initialize@all@0
1420  ⟨latexrelease⟩           \__hook_initialize_all:
1421  ⟨@=hook⟩
1422  ⟨latexrelease⟩ \EndIncludeInRelease

(End of 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.

```

1423  ⟨latexrelease⟩ \IncludeInRelease{2023/06/01}{\__hook_initialize_hook_code:n}
1424  ⟨latexrelease⟩           {Hooks~with~args}
1425  \cs_new_protected:Npn \__hook_initialize_hook_code:n #1
1426  {
1427    \__hook_debug:n
1428    { \iow_term:x { ^^J Update-code-for-hook~'##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.

```
1429  \__hook_include_legacy_code_chunk:n {#1}
```

If there aren't any code chunks for the current hook, there is no point in even starting the sorting routine so we make a quick test for that and in that case just update __hook_{hook} to hold the top-level and next code chunks. If there are code chunks we call __hook_initialize_single>NNn and pass to it ready made csnames as they are needed several times inside. This way we save a bit on processing time if we do that up front.

```

1430  \__hook_if_usable:nT {#1}
1431  {
1432    \prop_if_empty:cTF { g__hook##1_code_prop }
1433    {
```

```

1434     \_\_hook\_code\_gset:ne {#1}
1435     {

```

The hook may take arguments, so we add a run of braced parameters after the `_next` and `_toplevel` macros, so that the arguments passed to the hook are forwarded to them.

```

1436     \exp_not:c { __hook_toplevel~#1 } \_\_hook_braced_parameter:n {#1}
1437     \exp_not:c { __hook_next~#1 } \_\_hook_braced_parameter:n {#1}
1438     }
1439   }
1440   {

```

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.

```

1441     \_\_hook_if_reversed:nTF {#1}
1442     {
1443       \cs_set_eq:NN \_\_hook_tl_gput:Nn \_\_hook_tl_gput_left:Nn
1444       \cs_set_eq:NN \_\_hook_clist_gput:NV \clist_gput_left:NV }
1445       \cs_set_eq:NN \_\_hook_tl_gput:Nn \_\_hook_tl_gput_right:Nn
1446       \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.

```

1446     \prop_set_eq:Nc \l_\_hook_work_prop { g_\_hook_#1_code_prop }
1447     \_\_hook_initialize_single:ccn
1448     { __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.

```

1449     \_\_hook_debug:n
1450     {
1451       \exp_args:NNx \prop_gput:Nnn \g_\_hook_used_prop {#1} { } }
1452     }
1453   }
1454   \end{IncludeInRelease}
1455 \end{IncludeInRelease}{2020/10/01}{\_\_hook_initialize_hook_code:n}
1456 \end{IncludeInRelease} {Hooks-with-args}
1457 \cs_gset_protected:Npn \_\_hook_initialize_hook_code:n #1
1458 \end{IncludeInRelease} {
1459 \end{IncludeInRelease} \_\_hook_debug:n
1460 \end{IncludeInRelease} { \iow_term:x { ^J Update~code~for~hook~'#1' \on@line :^J } }
1461 \end{IncludeInRelease} \_\_hook_include_legacy_code_chunk:n {#1}
1462 \end{IncludeInRelease} \_\_hook_if_usable:nT {#1}
1463 \end{IncludeInRelease} {
1464 \end{IncludeInRelease} \prop_if_empty:cTF { g_\_hook_#1_code_prop }
1465 \end{IncludeInRelease} {
1466 \end{IncludeInRelease} \_\_hook_tl_gset:co { __hook~#1 }
1467 \end{IncludeInRelease} {
1468 \end{IncludeInRelease} \cs:w __hook_toplevel~#1 \exp_after:wN \cs_end:
1469 \end{IncludeInRelease} \cs:w __hook_next~#1 \cs_end:

```

```

1470 <|latexrelease>           }
1471 <|latexrelease>           }
1472 <|latexrelease>           }
1473 <|latexrelease>           }
1474 <|latexrelease>           }
1475 <|latexrelease>           }
1476 <|latexrelease>           }
1477 <|latexrelease>           }
1478 <|latexrelease>           }
1479 <|latexrelease>           }
1480 <|latexrelease>           }
1481 <|latexrelease>           }
1482 <|latexrelease>           }
1483 <|latexrelease>           }
1484 <|latexrelease>           }
1485 <|latexrelease>           }
1486 <|latexrelease> \EndIncludeInRelease

```

(End of 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.
`__hook_seq_csname:n`

FMi: note to myself: verify

```

1487 \cs_new:Npn \_\_hook_tl_csname:n #1 { l\_hook_label_#1_tl }
1488 \cs_new:Npn \_\_hook_seq_csname:n #1 { l\_hook_label_#1_seq }

```

(End of definition for `__hook_tl_csname:n` and `__hook_seq_csname:n.`)

```

\l\_hook_labels_seq
\l\_hook_labels_int
\l\_hook_front_tl
\l\_hook_rear_tl
\l\_hook_label_0_tl

```

For the sorting I am basically implementing Knuth's algorithm for topological sorting as given in TAOCP volume 1 pages 263–266. For this algorithm we need a number of local variables:

- List of labels used in the current hook to label code chunks:

```
1489 \seq_new:N \l\_hook_labels_seq
```

- Number of labels used in the current hook. In Knuth's algorithm this is called N :

```
1490 \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 :

```

1491 \tl_new:N \l\_hook_front_tl
1492 \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:

```
1493 \tl_new:c { \_\_hook_tl_csname:n { 0 } }
```

(End of definition for `\l__hook_labels_seq` and others.)

`__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 $\langle hook\text{-}code\text{-}plist \rangle$, $\langle hook\text{-}code\text{-}tl \rangle$, $\langle hook\text{-}top\text{-}level\text{-}code\text{-}tl \rangle$, $\langle hook\text{-}next\text{-}code\text{-}tl \rangle$, $\langle hook\text{-}ordered\text{-}labels\text{-}clist \rangle$ and $\langle hook\text{-}name \rangle$ (the latter is only used for debugging—the $\langle hook\text{-}rule\text{-}plist \rangle$ is accessed using the $\langle hook\text{-}name \rangle$).

The additional complexity compared to Don’s algorithm is that we do not use simple positive integers but have arbitrary alphanumeric labels. As usual Don’s data structures are chosen in a way that one can omit a lot of tests and I have mimicked that as far as possible. The result is a restriction I do not test for at the moment: a label can’t be equal to the number 0!

FMi: Needs checking for, just in case ... maybe

```
1494  ⟨latexrelease⟩ \IncludeInRelease{2023/06/01}{\__hook_initialize_single:NNn}
1495  ⟨latexrelease⟩
1496  \cs_new_protected:Npn \__hook_initialize_single:NNn #1#2#3
1497  {
```

Step T1: Initialize the data structure ...

```
1498  \seq_clear:N \l__hook_labels_seq
1499  \int_zero:N \l__hook_labels_int
```

Store the name of the hook:

```
1500  \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.

```
1501  \prop_map_inline:Nn \l__hook_work_prop
1502  {
1503      \int_incr:N \l__hook_labels_int
1504      \seq_put_right:Nn \l__hook_labels_seq {##1}
1505      \__hook_tl_set:cn { \__hook_tl_cname:n {##1} } { 0 }
1506      \seq_clear_new:c { \__hook_seq_cname:n {##1} }
1507 }
```

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.

```
1508  \prop_map_inline:Nn \l__hook_work_prop
1509  {
1510      \prop_map_inline:Nn \l__hook_work_prop
1511  {
1512      \__hook_if_label_case:nnnn {##1} {####1}
1513      { \prop_map_break: }
```

```

1514     { \__hook_apply_label_pair:nnn {##1} {####1} }
1515     { \__hook_apply_label_pair:nnn {####1} {##1} }
1516         {#3}
1517     }
1518 }
```

Now take a breath, and look at the data structures that have been set up:

```
1519     \__hook_debug:n { \__hook_debug_label_data:N \l__hook_work_prop }
```

Step T4:

```

1520     \tl_set:Nn \l__hook_rear_tl { 0 }
1521     \tl_set:cn { \__hook_tl_cname:n { 0 } } { 0 }
1522     \seq_map_inline:Nn \l__hook_labels_seq
1523     {
1524         \int_compare:nNnT { \cs:w \__hook_tl_cname:n {##1} \cs_end: } = 0
1525             {
1526                 \tl_set:cn { \__hook_tl_cname:n { \l__hook_rear_tl } }{##1}
1527                 \tl_set:Nn \l__hook_rear_tl {##1}
1528             }
1529         }
1530     \tl_set_eq:Nc \l__hook_front_tl { \__hook_tl_cname:n { 0 } }
1531     \__hook_tl_gclear:N #1
1532     \clist_gclear:N #2
```

The whole loop gets combined in steps T5–T7:

```
1533     \bool_while_do:nn { ! \str_if_eq_p:Vn \l__hook_front_tl { 0 } }
1534     {
```

This part is step T5:

```

1535     \int_decr:N \l__hook_labels_int
1536     \prop_get:NVN \l__hook_work_prop \l__hook_front_tl \l__hook_return_tl
1537     \exp_args:NNV \__hook_tl_gput:Nn #1 \l__hook_return_tl
1538     \__hook_clist_gput:NV #2 \l__hook_front_tl
1539     \__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.

```

1540     \seq_map_inline:cn { \__hook_seq_cname:n { \l__hook_front_tl } }
1541     {
1542         \tl_set:cx { \__hook_tl_cname:n {##1} }
1543             {
1544                 \int_eval:n
1545                     { \cs:w \__hook_tl_cname:n {##1} \cs_end: - 1 }
1546             }
1547         \int_compare:nNnT
1548             { \cs:w \__hook_tl_cname:n {##1} \cs_end: } = 0
1549             {
1550                 \tl_set:cn { \__hook_tl_cname:n { \l__hook_rear_tl } }{##1}
1551                 \tl_set:Nn \l__hook_rear_tl {##1}
1552             }
1553     }
```

and here is step T7:

```
1553     \tl_set_eq:Nc \l__hook_front_tl
1554             { \__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.

```

1555     }
1556     \int_compare:nNnF \l__hook_labels_int = 0
1557     {
1558         \iow_term:x{=====}
1559         \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

```

1560     \__hook_debug_label_data:N \l__hook_work_prop
1561     \iow_term:x{=====}
1562 }

```

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 (to the right). The hook may take arguments, so we add a run of braced parameters after the `_next` and `_toplevel` macros, so that the arguments passed to the hook are forwarded to them.

```

1563 \exp_args:NN \__hook_tl_gput:Nn #1
1564     { \exp_not:c { __hook_toplevel~#3 } \__hook_braced_parameter:n {#3} }
1565 \__hook_tl_gput_right:Ne #1
1566     { \exp_not:c { __hook_next~#3 } \__hook_braced_parameter:n {#3} }
1567 \use:e
1568     {
1569         \cs_gset:cpn { __hook~#3 } \use:c { c__hook_#3_parameter_tl }
1570         { \exp_not:V #1 }
1571     }
1572 }

1573 \cs_generate_variant:Nn \__hook_initialize_single:NNn { cc }
1574 \end{IncludeInRelease}

1575 \IncludeInRelease{2020/10/01}{\__hook_initialize_single:NNn}
1576 \begin{Hooks-with-args}
1577 \cs_new_protected:Npn \__hook_initialize_single:NNn #1#2#3
1578 \end{Hooks-with-args}
1579 \seq_clear:N \l__hook_labels_seq
1580 \int_zero:N \l__hook_labels_int
1581 \tl_set:Nn \l__hook_cur_hook_tl {#3}
1582 \prop_map_inline:Nn \l__hook_work_prop
1583 \seq_clear_new:c { \__hook_seq_cname:n {##1} } { 0 }
1584 \int_incr:N \l__hook_labels_int
1585 \seq_put_right:Nn \l__hook_labels_seq {##1}
1586 \__hook_tl_set:cn { \__hook_tl_cname:n {##1} } { 0 }
1587 \seq_clear_new:c { \__hook_seq_cname:n {##1} }
1588 \prop_map_inline:Nn \l__hook_work_prop
1589 \prop_map_inline:Nn \l__hook_work_prop
1590 \prop_map_inline:Nn \l__hook_work_prop
1591 \end{IncludeInRelease}

```

```

1592 <|latexrelease>
1593 <|latexrelease>
1594 <|latexrelease>
1595 <|latexrelease>
1596 <|latexrelease>
1597 <|latexrelease>
1598 <|latexrelease>
1599 <|latexrelease>
1600 <|latexrelease>
1601 <|latexrelease>
1602 <|latexrelease>
1603 <|latexrelease>
1604 <|latexrelease>
1605 <|latexrelease>
1606 <|latexrelease>
1607 <|latexrelease>
1608 <|latexrelease>
1609 <|latexrelease>
1610 <|latexrelease>
1611 <|latexrelease>
1612 <|latexrelease>
1613 <|latexrelease>
1614 <|latexrelease>
1615 <|latexrelease>
1616 <|latexrelease>
1617 <|latexrelease>
1618 <|latexrelease>
1619 <|latexrelease>
1620 <|latexrelease>
1621 <|latexrelease>
1622 <|latexrelease>
1623 <|latexrelease>
1624 <|latexrelease>
1625 <|latexrelease>
1626 <|latexrelease>
1627 <|latexrelease>
1628 <|latexrelease>
1629 <|latexrelease>
1630 <|latexrelease>
1631 <|latexrelease>
1632 <|latexrelease>
1633 <|latexrelease>
1634 <|latexrelease>
1635 <|latexrelease>
1636 <|latexrelease>
1637 <|latexrelease>
1638 <|latexrelease>
1639 <|latexrelease>
1640 <|latexrelease>
1641 <|latexrelease>
1642 <|latexrelease>
1643 <|latexrelease>
1644 <|latexrelease>
1645 <|latexrelease>

{
    \__hook_if_label_case:nnnn {##1} {####1}
    { \prop_map_break: }
    { \__hook_apply_label_pair:nnn {##1} {####1} }
    { \__hook_apply_label_pair:nnn {##1} {####1} }
    {##3}
}

\__hook_debug:n { \__hook_debug_label_data:N \l__hook_work_prop }
\tl_set:Nn \l__hook_rear_tl { 0 }
\tl_set:cn { \__hook_tl_cname:n { 0 } } { 0 }
\seq_map_inline:Nn \l__hook_labels_seq
{
    \int_compare:nNnT { \cs:w \__hook_tl_cname:n {##1} \cs_end: } = 0
    {
        \tl_set:cn { \__hook_tl_cname:n { \l__hook_rear_tl } }{##1}
        \tl_set:Nn \l__hook_rear_tl {##1}
    }
}
\__hook_set_eq:Nc \l__hook_front_tl { \__hook_tl_cname:n { 0 } }
\__hook_tl_gclear:N #1
\clist_gclear:N #2
\bool_while_do:nn { ! \str_if_eq_p:Vn \l__hook_front_tl { 0 } }
{
    \int_decr:N \l__hook_labels_int
    \prop_get:NVN \l__hook_work_prop \l__hook_front_tl \l__hook_return_tl
    \exp_args:NNV \__hook_tl_gput:Nn #1 \l__hook_return_tl
    \__hook_clist_gput:NV #2 \l__hook_front_tl
    \__hook_debug:n{ \iow_term:x{Handled~ code~ for~ \l__hook_front_tl} }
    \seq_map_inline:cn { \__hook_seq_cname:n { \l__hook_front_tl } }
    {
        \tl_set:cx { \__hook_tl_cname:n {##1} }
        { \int_eval:n
            { \cs:w \__hook_tl_cname:n {##1} \cs_end: - 1 }
        }
    }
    \int_compare:nNnt
    { \cs:w \__hook_tl_cname:n {##1} \cs_end: } = 0
    {
        \tl_set:cn { \__hook_tl_cname:n { \l__hook_rear_tl } }{##1}
        \tl_set:Nn \l__hook_rear_tl {##1}
    }
}
\__hook_set_eq:Nc \l__hook_front_tl
{ \__hook_tl_cname:n { \l__hook_front_tl } }
\int_compare:nNnF \l__hook_labels_int = 0
{
    \iow_term:x{=====}
    \iow_term:x{Error:~ label~ rules~ are~ incompatible:}
    \__hook_debug_label_data:N \l__hook_work_prop
    \iow_term:x{=====}
}
\exp_args:NNo \__hook_tl_gput:Nn #1 { \cs:w __hook_toplevel~#3 \cs_end: }
\__hook_tl_gput_right:Nn #1 { \cs:w __hook_next~#3 \cs_end: }

```

```

1646  \langle latexrelease\rangle    }
1647  \langle latexrelease\rangle \cs_generate_variant:Nn \__hook_tl_gput_right:Nn { No }
1648  \langle latexrelease\rangle \EndIncludeInRelease

```

(End of definition for `__hook_initialize_single:Nnn`.)

`__hook_tl_gput:Nn` `__hook_clist_gput:NV` These append either on the right (normal hook) or on the left (reversed hook). This is setup up in `__hook_initialize_hook_code:n`, elsewhere their behavior is undefined.

```

1649  \cs_new:Npn \__hook_tl_gput:Nn { \ERROR }
1650  \cs_new:Npn \__hook_clist_gput:NV { \ERROR }

```

(End of definition for `__hook_tl_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`.

```

1651  \cs_new_protected:Npn \__hook_apply_label_pair:nnn #1#2#3
1652  {

```

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.

```

1653      \__hook_label_if_exist_apply:nnnF {#1} {#2} {#3}
1654      {

```

If there is no hook-specific rule we check for a default one and use that if it exists.

```

1655          \__hook_label_if_exist_apply:nnnF {#1} {#2} {??} {
1656      }
1657  }
1658  \cs_new_protected:Npn \__hook_label_if_exist_apply:nnnF #1#2#3
1659  {
1660      \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`.

```

1661      \__hook_apply_rule:nnn {#1} {#2} {#3}
1662          \exp_after:wN \use_none:n
1663      \else:
1664          \use:nn
1665      \fi:
1666  }

```

(End of 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 $\langle label1 \rangle$, $\langle label2 \rangle$, $\langle hook-name \rangle$, and $\langle hook-code-plist \rangle$.

```

1667 \cs_new_protected:Npn \_\_hook_apply_rule:nnn #1#2#3
1668 {
1669   \cs:w __hook_apply_
1670   \cs:w g__hook_#3_reversed_tl \cs_end: rule_
1671   \cs:w g__hook_ #3 _rule_ #1 | #2 _tl \cs_end: :nnn \cs_end:
1672   {#1} {#2} {#3}
1673 }

```

(End of 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.

```

1674 \cs_new_protected:cpn { __hook_apply_rule_<:nnn } #1#2#3
1675 {
1676   __hook_debug:n { __hook_msg_pair_found:nnn {#1} {#2} {#3} }
1677   \tl_set:cx { __hook_tl_curname:n {#2} }
1678   { \int_eval:n{ \cs:w __hook_tl_curname:n {#2} \cs_end: + 1 } }
1679   \seq_put_right:cnf { __hook_seq_curname:n {#1} }{#2}
1680 }
1681 \cs_new_protected:cpn { __hook_apply_rule_>:nnn } #1#2#3
1682 {
1683   __hook_debug:n { __hook_msg_pair_found:nnn {#1} {#2} {#3} }
1684   \tl_set:cx { __hook_tl_curname:n {#1} }
1685   { \int_eval:n{ \cs:w __hook_tl_curname:n {#1} \cs_end: + 1 } }
1686   \seq_put_right:cnf { __hook_seq_curname:n {#2} }{#1}
1687 }

```

(End of 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.

```

1688 \cs_new_protected:cpn { __hook_apply_rule_xE:nnn } #1#2#3
1689 {
1690   __hook_debug:n { __hook_msg_pair_found:nnn {#1} {#2} {#3} }
1691   \msg_error:nnnnn { hooks } { labels-incompatible }
1692   {#1} {#2} {#3} { 1 }
1693   \use:c { __hook_apply_rule_->:nnn } {#1} {#2} {#3}
1694   \use:c { __hook_apply_rule_-<:nnn } {#1} {#2} {#3}
1695 }
1696 \cs_new_protected:cpn { __hook_apply_rule_xW:nnn } #1#2#3
1697 {
1698   __hook_debug:n { __hook_msg_pair_found:nnn {#1} {#2} {#3} }
1699   \msg_warning:nnnnn { hooks } { labels-incompatible }
1700   {#1} {#2} {#3} { 0 }
1701 }

```

(End of 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.¹¹ So we still unnecessarily try to sort it in and depending on the rules that might result in a loop that is otherwise resolved. If that turns out to be a real issue, we can improve the code.

Here the code is removed from `\l__hook_cur_hook_t1` 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_t1` makes the default rules -> and <- work properly.

```

1702 \cs_new_protected:cpn { __hook_apply_rule_->:nnn } #1#2#3
1703 {
1704     __hook_debug:n
1705     {
1706         __hook_msg_pair_found:nnn {#1} {#2} {#3}
1707         \iow_term:x{--->~ Drop~ '#2'~ code~ from~
1708             \iow_char:N \\ g_hook_ \l__hook_cur_hook_t1 _code_prop ~
1709             because~ of~ '#1' }
1710     }
1711     \prop_put:Nnn \l__hook_work_prop {#2} { }
1712 }
1713 \cs_new_protected:cpn { __hook_apply_rule_-<:nnn } #1#2#3
1714 {
1715     __hook_debug:n
1716     {
1717         __hook_msg_pair_found:nnn {#1} {#2} {#3}
1718         \iow_term:x{--->~ Drop~ '#1'~ code~ from~
1719             \iow_char:N \\ g_hook_ \l__hook_cur_hook_t1 _code_prop ~
1720             because~ of~ '#2' }
1721     }
1722     \prop_put:Nnn \l__hook_work_prop {#1} { }
1723 }
```

(End of definition for `__hook_apply_rule_->:nnn` and `__hook_apply_rule_-<:nnn`.)

`__hook_apply_-rule_-<:nnn`

Reversed rules.

```

1724 \cs_new_eq:cc { __hook_apply_-rule_-<:nnn } { __hook_apply_rule_->:nnn }
1725 \cs_new_eq:cc { __hook_apply_-rule_->:nnn } { __hook_apply_rule_-<:nnn }
1726 \cs_new_eq:cc { __hook_apply_-rule_-<:nnn } { __hook_apply_rule_-<:nnn }
1727 \cs_new_eq:cc { __hook_apply_-rule_->:nnn } { __hook_apply_rule_->:nnn }
1728 \cs_new_eq:cc { __hook_apply_-rule_xE:nnn } { __hook_apply_rule_xE:nnn }
1729 \cs_new_eq:cc { __hook_apply_-rule_xW:nnn } { __hook_apply_rule_xW:nnn }
```

(End of definition for `__hook_apply_-rule_-<:nnn` and others.)

`__hook_msg_pair_found:nnn`

A macro to avoid moving this many tokens around.

```

1730 \cs_new_protected:Npn \__hook_msg_pair_found:nnn #1#2#3
1731 {
1732     \iow_term:x{~ \str_if_eq:nnTF {#3} {??} {default} {~normal} ~
1733         rule~ \__hook_label_pair:nn {#1} {#2}:~}
1734         \use:c { g_hook_#3_rule_ \__hook_label_pair:nn {#1} {#2} _t1 } ~
1735         found}
1736 }
```

(End of definition for `__hook_msg_pair_found:nnn`.)

¹¹This 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.

```

\_\_hook_debug_label_data:N
1737 \cs_new_protected:Npn \_\_hook_debug_label_data:N #1 {
1738   \iow_term:x{Code~ labels~ for~ sorting:}
1739   \iow_term:x{~ \seq_use:Nnnn\l_\_hook_labels_seq {~and~}{,~}{~and~} }
1740   \iow_term:x{^\^\J Data~ structure~ for~ label~ rules:}
1741   \prop_map_inline:Nn #1
1742   {
1743     \iow_term:x{~ ##1~ == \tl_use:c{ \_\_hook_tl_cname:n {##1} }~ ->~
1744       \seq_use:cnnn{ \_\_hook_seq_cname:n {##1} }{~->}{}{~->~}{~->~}
1745     }
1746   }
1747   \iow_term:x{}
1748 }

(End of 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_line:x
\_\_hook_log_line_indent:x
\_\_hook_log:nN
1749 \cs_new_protected:Npn \hook_log:n #1
1750   {
1751     \cs_set_eq:NN \_\_hook_log_cmd:x \iow_log:x
1752     \_\_hook_normalize_hook_args:Nn \_\_hook_log:nN {#1} \tl_log:x
1753   }
1754 \cs_new_protected:Npn \hook_show:n #1
1755   {
1756     \cs_set_eq:NN \_\_hook_log_cmd:x \iow_term:x
1757     \_\_hook_normalize_hook_args:Nn \_\_hook_log:nN {#1} \tl_show:x
1758   }
1759 \cs_new_protected:Npn \_\_hook_log_line:x #1
1760   { \_\_hook_log_cmd:x { >~#1 } }
1761 \cs_new_protected:Npn \_\_hook_log_line_indent:x #1
1762   { \_\_hook_log_cmd:x { >~\Cspaces #1 } }

\!\!(\!\! latexrelease \!\!) \IncludeInRelease{2023/06/01}{\_\_hook_log:nN}
\!\!(\!\! latexrelease \!\!) \!\! {Hooks~with~args}
1765 \cs_new_protected:Npn \_\_hook_log:nN #1 #2
1766   {
1767     \_\_hook_if_deprecated_generic:nT {#1}
1768     {
1769       \_\_hook_DEPRECATED_GENERIC_WARN:n {#1}
1770       \_\_hook_do_DEPRECATED_GENERIC:Nn \_\_hook_log:nN {#1} #2
1771       \exp_after:wN \use:none:nnnnnnnn \use:none:nnnn
1772     }
1773     \_\_hook_preamble_hook:n {#1}
1774     \_\_hook_log_cmd:x
1775     {
1776       ^^\J ->~The~
1777       \_\_hook_if_generic:nT {#1} { generic~ }
1778       hook~'#1'
1779       \_\_hook_if_disabled:nF {#1}
1780       {
1781         \exp_args:Nf \_\_hook_print_args:nn {#1}
1782       }

```

```

1783         \int_eval:n
1784             { \str_count:e { \__hook_parameter:n {#1} } / 3 }
1785         }
1786     }
1787     :
1788 }
1789 \__hook_if_usable:nF {#1}
1790     { \__hook_log_line:x { The~hook~is~not~declared. } }
1791 \__hook_if_disabled:nT {#1}
1792     { \__hook_log_line:x { The~hook~is~disabled. } }
1793 \hook_if_empty:nTF {#1}
1794     { #2 { The~hook~is~empty } }
1795 {
1796     \__hook_log_line:x { Code~chunks: }
1797     \prop_if_empty:cTF { g__hook_#1_code_prop }
1798         { \__hook_log_line_indent:x { --- } }
1799         {
1800             \prop_map_inline:cn { g__hook_#1_code_prop }
1801             {
1802                 \exp_after:wN \cs_set:Npn \exp_after:wN \__hook_tmp:w
1803                     \c__hook_nine_parameters_tl {##2}
1804                     \__hook_log_line_indent:x
1805                         { ##1~->~\cs_replacement_spec:N \__hook_tmp:w }
1806             }
1807         }

```

If there is code in the top-level token list, print it:

```

1808 \__hook_log_line:x
1809 {
1810     Document-level~(top-level)~code
1811     \__hook_if_usable:nT {#1}
1812         { ~(executed~\__hook_if_reversed:nTF {#1} {first} {last} ) } :
1813     }
1814     \__hook_log_line_indent:x
1815     {
1816         \__hook_cs_if_empty:cTF { __hook_toplevel~#1 }
1817             { --- }
1818             { -> ~ \cs_replacement_spec:c { __hook_toplevel~#1 } }
1819     }
1820     \__hook_log_line:x { Extra~code~for~next~invocation: }
1821     \__hook_log_line_indent:x
1822     {
1823         \__hook_cs_if_empty:cTF { __hook_next~#1 }
1824             { --- }

```

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.

```

1825 {
1826     -> ~ \exp_last_unbraced:Nf \__hook_log_next_code:w
1827         { \cs_replacement_spec:c { __hook_next~#1 } }
1828     }
1829 }
```

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.

```

1830     \__hook_log_line:x { Rules: }
1831     \bool_set_true:N \l__hook_tmpa_bool
1832     \__hook_list_rules:nn {#1}
1833     {
1834         \bool_set_false:N \l__hook_tmpa_bool
1835         \__hook_log_line_indent:x
1836         {
1837             ##2~ with~
1838             \str_if_eq:nnT {##3} {??} { default~ }
1839             relation~ ##1
1840         }
1841     }
1842     \bool_if:NT \l__hook_tmpa_bool
1843     { \__hook_log_line_indent:x { --- } }

```

When the hook is declared (that is, the sorting algorithm is applied to that hook) and not empty

```

1844     \bool_lazy_and:nnTF
1845     { \__hook_if_usable_p:n {#1} }
1846     { ! \hook_if_empty_p:n {#1} }
1847     {
1848         \__hook_log_line:x
1849         {
1850             Execution~order
1851             \bool_if:NTF \l__hook_tmpa_bool
1852             { \__hook_if_reversed:nT {#1} { ~(after~reversal) } }
1853             { ~(after~
1854                 \__hook_if_reversed:nT {#1} { reversal~and~ }
1855                 applying~rules)
1856             } :
1857         }
1858         #2 % \tl_show:n
1859         {
1860             \@spaces
1861             \clist_if_empty:cTF { g__hook_#1_labels_clist }
1862             { --- }
1863             { \clist_use:cn { g__hook_#1_labels_clist } { ,~ } }
1864         }
1865     }
1866     {
1867         \__hook_log_line:x { Execution~order: }
1868         #2
1869         {
1870             \@spaces Not~set~because~the~hook~ \__hook_if_usable:nTF {#1}
1871             { code~pool~is~empty }
1872             { is~\__hook_if_disabled:nTF {#1} {disabled} {undeclared} }
1873         }
1874     }
1875 }
1876 }
1877 \! latexrelease \EndIncludeInRelease

```

```

1878 %
1879 <|latexrelease>\IncludeInRelease{2020/10/01}{\_\_hook\_log:nN}
1880 <|latexrelease>                                {Hooks~with~args}
1881 <|latexrelease>\cs_new_protected:Npn \_\_hook\_log:nN #1 #2
1882 <|latexrelease> {
1883 <|latexrelease>   \_\_hook_if_deprecated_generic:nT {#1}
1884 <|latexrelease>   {
1885 <|latexrelease>     \_\_hook_DEPRECATED_GENERIC_WARN:n {#1}
1886 <|latexrelease>     \_\_hook_DO_DEPRECATED_GENERIC:Nn \_\_hook_log:nN {#1} #2
1887 <|latexrelease>     \exp_after:wN \use_none:nnnnnnnn \use_none:nnnn
1888 <|latexrelease>
1889 <|latexrelease> \_\_hook_preamble_hook:n {#1}
1890 <|latexrelease> \_\_hook_log_cmd:x
1891 <|latexrelease>   { ^J ->~ The~ \_\_hook_if_generic:nT {#1} { generic~ } hook~'#1': }
1892 <|latexrelease> \_\_hook_if_usable:nF {#1}
1893 <|latexrelease>   { \_\_hook_log_line:x { The~hook~is~not~declared. } }
1894 <|latexrelease> \_\_hook_if_disabled:nT {#1}
1895 <|latexrelease>   { \_\_hook_log_line:x { The~hook~is~disabled. } }
1896 <|latexrelease> \hook_if_empty:nTF {#1}
1897 <|latexrelease>   { #2 { The~hook~is~empty } }
1898 <|latexrelease>
1899 <|latexrelease>   {
1900 <|latexrelease>     \_\_hook_log_line:x { Code~chunks: }
1901 <|latexrelease>     \prop_if_empty:cTF { g_\_\_hook_\#1_code_prop }
1902 <|latexrelease>       { \_\_hook_log_line_indent:x { --- } }
1903 <|latexrelease>       {
1904 <|latexrelease>         \prop_map_inline:cn { g_\_\_hook_\#1_code_prop }
1905 <|latexrelease>           { \_\_hook_log_line_indent:x { ##1~->-\tl_to_str:n {##2} } }
1906 <|latexrelease>
1907 <|latexrelease>   \_\_hook_log_line:x
1908 <|latexrelease>   {
1909 <|latexrelease>     Document-level~(top-level)~code
1910 <|latexrelease>     \_\_hook_if_usable:nT {#1}
1911 <|latexrelease>       { ~ (executed~\_\_hook_if_reversed:nTF {#1} {first} {last} ) : }
1912 <|latexrelease>
1913 <|latexrelease>   \_\_hook_log_line_indent:x
1914 <|latexrelease>   {
1915 <|latexrelease>     \tl_if_empty:cTF { \_\_hook_toplevel~#1 }
1916 <|latexrelease>       { --- }
1917 <|latexrelease>       { -> ~ \exp_args:Nv \tl_to_str:n { \_\_hook_toplevel~#1 } }
1918 <|latexrelease>
1919 <|latexrelease> \_\_hook_log_line_indent:x
1920 <|latexrelease>
1921 <|latexrelease>   {
1922 <|latexrelease>     \tl_if_empty:cTF { \_\_hook_next~#1 }
1923 <|latexrelease>       { --- }
1924 <|latexrelease>       { ->~ \exp_args:Nv \_\_hook_log_next_code:n { \_\_hook_next~#1 } }
1925 <|latexrelease>
1926 <|latexrelease> \bool_set_true:N \l_\_\_hook_tmpt_bool
1927 <|latexrelease> \_\_hook_list_rules:nn {#1}
1928 <|latexrelease>
1929 <|latexrelease> \bool_set_false:N \l_\_\_hook_tmpt_bool
1930 <|latexrelease> \_\_hook_log_line_indent:x
1931 <|latexrelease>   {

```

```

1932  ⟨latexrelease⟩          ##2~ with~
1933  ⟨latexrelease⟩          \str_if_eq:nnT {##3} {??} { default~ }
1934  ⟨latexrelease⟩          relation~ ##1
1935  ⟨latexrelease⟩          }
1936  ⟨latexrelease⟩          }
1937  ⟨latexrelease⟩          \bool_if:NT \l__hook_tmpa_bool
1938  ⟨latexrelease⟩          { \__hook_log_line_indent:x { --- } }
1939  ⟨latexrelease⟩          \bool_lazy_and:nnTF
1940  ⟨latexrelease⟩          { \__hook_if_usable_p:n {#1} }
1941  ⟨latexrelease⟩          { ! \hook_if_empty_p:n {#1} }
1942  ⟨latexrelease⟩          {
1943  ⟨latexrelease⟩          \__hook_log_line:x
1944  ⟨latexrelease⟩          {
1945  ⟨latexrelease⟩          Execution~order
1946  ⟨latexrelease⟩          \bool_if:NTF \l__hook_tmpa_bool
1947  ⟨latexrelease⟩          { \__hook_if_reversed:nT {#1} { ~(after~reversal) } }
1948  ⟨latexrelease⟩          { ~(after~
1949  ⟨latexrelease⟩          \__hook_if_reversed:nT {#1} { reversal~and~ }
1950  ⟨latexrelease⟩          applying~rules)
1951  ⟨latexrelease⟩          } :
1952  ⟨latexrelease⟩          }
1953  ⟨latexrelease⟩          #2 % \tl_show:n
1954  ⟨latexrelease⟩          {
1955  ⟨latexrelease⟩          \cclist_if_empty:cTF { g__hook_#1_labels_clist }
1956  ⟨latexrelease⟩          { --- }
1957  ⟨latexrelease⟩          { \cclist_use:cn { g__hook_#1_labels_clist } { ,~ } }
1958  ⟨latexrelease⟩          }
1959  ⟨latexrelease⟩          }
1960  ⟨latexrelease⟩          }
1961  ⟨latexrelease⟩          }
1962  ⟨latexrelease⟩          }
1963  ⟨latexrelease⟩          }
1964  ⟨latexrelease⟩          }
1965  ⟨latexrelease⟩          \__hook_log_line:x { Execution~order: }
1966  ⟨latexrelease⟩          #2
1967  ⟨latexrelease⟩          {
1968  ⟨latexrelease⟩          \cspaces Not~set~because~the~hook~ \__hook_if_usable:nTF {#1}
1969  ⟨latexrelease⟩          { code~pool~is~empty }
1970  ⟨latexrelease⟩          { is~\__hook_if_disabled:nTF {#1} {disabled} {undeclared} }
1971  ⟨latexrelease⟩          }
1972  ⟨latexrelease⟩          }
1973  ⟨latexrelease⟩ \EndIncludeInRelease{2023/06/01}{\__hook_log_next_code:n}
1974  ⟨latexrelease⟩          {Hooks~with~args}
\__hook_log_next_code:n
1975  \exp_last_unbraced:NNNNo
1976  \cs_new:Npn \__hook_log_next_code:w #1 \c_right_brace_str { }
1977  ⟨latexrelease⟩ \EndIncludeInRelease
1978  ⟨latexrelease⟩ \EndIncludeInRelease{2020/10/01}{\__hook_log_next_code:n}
1979  ⟨latexrelease⟩          {Hooks~with~args}
1980  ⟨latexrelease⟩ \cs_gset:Npn \__hook_log_next_code:n #1
1981  ⟨latexrelease⟩          { \exp_args:No \tl_to_str:n { \use_none:nn #1 } }

```

To display the code for next invocation only (i.e., from \AddToHookNext we have to remove the string __hook_clear_next:n{\hook}, so the simplest is to use a macro delimited by a }12.

```

1973  ⟨latexrelease⟩ \EndIncludeInRelease{2023/06/01}{\__hook_log_next_code:n}
1974  ⟨latexrelease⟩          {Hooks~with~args}
\__hook_log_next_code:n
1975  \exp_last_unbraced:NNNNo
1976  \cs_new:Npn \__hook_log_next_code:w #1 \c_right_brace_str { }
1977  ⟨latexrelease⟩ \EndIncludeInRelease
1978  ⟨latexrelease⟩ \EndIncludeInRelease{2020/10/01}{\__hook_log_next_code:n}
1979  ⟨latexrelease⟩          {Hooks~with~args}
1980  ⟨latexrelease⟩ \cs_gset:Npn \__hook_log_next_code:n #1
1981  ⟨latexrelease⟩          { \exp_args:No \tl_to_str:n { \use_none:nn #1 } }

```

```
1982  \cs_new:Npn \__hook_print_args:nn #1 #2
```

Pretty-prints the number of arguments of a hook.

```
1983  \cs_new:Npn \__hook_print_args:nn #1 #2
1984  {
1985      \int_compare:nNnT {#2} > { 0 }
1986      {
1987          \__hook_if_declared:nT {#1} { \use_none:nnn }
1988          \__hook_if_cmd_hook:nT {#1}
1989          { \use_i:nnn { ~ (unknown ~ ) } }
1990          \use:n { ~ (#2 ~ ) }
1991          argument \int_compare:nNnT {#2} > { 1 } { s }
1992      }
1993  }
```

(End of definition for `\hook_show:n` and others. These functions are documented on page 206.)

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).

```
1994 \cs_new_protected:Npn \__hook_list_rules:nn #1 #2
1995  {
1996      \cs_set_protected:Npn \__hook_tmp:w ##1 ##2 ##3 {#2}
1997      \prop_map_inline:cn { g__hook_#1_code_prop }
1998      {
1999          \prop_map_inline:cn { g__hook_#1_code_prop }
2000          {
2001              \__hook_if_label_case:nnnn {##1} {####1}
2002              { \prop_map_break: }
2003              { \__hook_list_one_rule:nnn {##1} {####1} }
2004              { \__hook_list_one_rule:nnn {####1} {##1} }
2005              {##1}
2006          }
2007      }
2008  }
```

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>`.

```
2009 \cs_new_protected:Npn \__hook_list_one_rule:nnn #1#2#3
2010  {
2011      \__hook_list_if_rule_exists:nnnF {#1} {#2} {#3}
2012      { \__hook_list_if_rule_exists:nnnF {#1} {#2} { ?? } { } }
2013  }
2014 \cs_new_protected:Npn \__hook_list_if_rule_exists:nnnF #1#2#3
2015  {
2016      \if_cs_exist:w g__hook_ #3 _rule_ #1 | #2 _tl \cs_end:
2017      \exp_args:Nv \__hook_tmp:w
2018      { g__hook_ #3 _rule_ #1 | #2 _tl } { #1 | #2 } {#3}
2019      \exp_after:wN \use_none:nn
2020  \fi:
2021  \use:n
2022  }
```

(End of 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`.

```

2023 \cs_new_protected:Npn \_hook_debug_print_rules:n #1
2024 {
2025   \iow_term:n { The~hook~#1~contains~the~rules: }
2026   \cs_set_protected:Npn \_hook_tmp:w ##1
2027   {
2028     \_hook_list_rules:nn {#1}
2029     {
2030       \iow_term:x
2031       {
2032         > ##1 {####2} ##1 => ##1 {####1}
2033         \str_if_eq:nnT {####3} {??} { ~ (default) }
2034       }
2035     }
2036   }
2037   \exp_args:No \_hook_tmp:w { \use:nn { ~ } { ~ } }
2038 }
```

(End of definition for `_hook_debug_print_rules:n`.)

4.8 Specifying code for next invocation

`\hook_gput_next_code:nn`

```

2039 ⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\hook_gput_next_code:nn}
2040 ⟨latexrelease⟩
2041   {Hooks~with~args}
2041 \cs_new_protected:Npn \hook_gput_next_code:nn #1 #2
2042   {
2043     \_hook_replacing_args_false:
2044     \_hook_normalize_hook_args:Nn \_hook_gput_next_code:nn {#1} {#2}
2045     \_hook_replacing_args_reset:
2046   }
2047 \cs_new_protected:Npn \hook_gput_next_code_with_args:nn #1 #2
2048   {
2049     \_hook_replacing_args_true:
2050     \_hook_normalize_hook_args:Nn \_hook_gput_next_code:nn {#1} {#2}
2051     \_hook_replacing_args_reset:
2052   }
2053 ⟨latexrelease⟩\EndIncludeInRelease
2054 ⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\hook_gput_next_code:nn}
2055 ⟨latexrelease⟩
2055   {Hooks~with~args}
2056 ⟨latexrelease⟩\cs_gset_protected:Npn \hook_gput_next_code:nn #1
2057 ⟨latexrelease⟩ { \_hook_normalize_hook_args:Nn \_hook_gput_next_code:nn {#1} }
2058 ⟨latexrelease⟩\cs_gset_protected:Npn \hook_gput_next_code_with_args:nn #1 #2 { }
2059 ⟨latexrelease⟩\EndIncludeInRelease
```

(End of definition for `\hook_gput_next_code:nn`. This function is documented on page 205.)

`_hook_gput_next_code:nn`

```

2060 \cs_new_protected:Npn \_hook_gput_next_code:nn #1 #2
2061   {
2062     \_hook_if_disabled:nTF {#1}
```

```

2063     { \msg_error:n { hooks } { hook-disabled } {#1} }
2064     {
2065         \__hook_if_structure_exist:nTF {#1}
2066         { \__hook_gput_next_do:nn
2067             { \__hook_try_declar ing_generic_next_hook:nn }
2068             {#1} {#2}
2069         }
2070     }

```

(End of definition for `__hook_gput_next_code:nn`.)

`__hook_gput_next_do:nn` Start by sanity-checking with `__hook_chk_args_allowed:nn`. Then 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.

```

2071 \langle latexrelease \rangle \IncludeInRelease{2023/06/01}{\__hook_gput_next_do:nn}
2072 \langle latexrelease \rangle
2073 \cs_new_protected:Npn \__hook_gput_next_do:nn #1
2074 {
2075     \__hook_init_structure:n {#1}
2076     \__hook_chk_args_allowed:nn {#1} { AddToHookNext }
2077     \__hook_cs_if_empty:cT { __hook~#1 }
2078     { \__hook_update_hook_code:n {#1} }
2079     \__hook_cs_if_empty:cT { __hook_next~#1 }
2080     { \__hook_next_gset:nn {#1} { \__hook_clear_next:n {#1} } }
2081     \__hook_cs_gput_right:nnn { _next } {#1}
2082 }
2083 \langle latexrelease \rangle \EndIncludeInRelease
2084 \langle latexrelease \rangle \IncludeInRelease{2020/10/01}{\__hook_gput_next_do:nn}
2085 \langle latexrelease \rangle
2086 \cs_gset_protected:Npn \__hook_gput_next_do:nn #1
2087 \langle latexrelease \rangle
2088 \exp_args:Nc \__hook_gput_next_do:Nnn
2089 { __hook_next~#1 } {#1}
2090 \langle latexrelease \rangle
2091 \cs_gset_protected:Npn \__hook_gput_next_do:Nnn #1 #2
2092 \langle latexrelease \rangle
2093 \tl_if_empty:cT { __hook~#2 }
2094 { \__hook_update_hook_code:n {#2} }
2095 \tl_if_empty:NT #1
2096 { \__hook_tl_gset:Nn #1 { \__hook_clear_next:n {#2} } }
2097 \__hook_tl_gput_right:Nn #1
2098 \langle latexrelease \rangle
2099 \EndIncludeInRelease

```

(End of definition for `__hook_gput_next_do:nn`.)

`\hook_gclear_next_code:n` Discard anything set up for next invocation of the hook.

```

2100 \cs_new_protected:Npn \hook_gclear_next_code:n #1
2101 { \__hook_normalize_hook_args:Nn \__hook_clear_next:n {#1} }

```

(End of definition for `\hook_gclear_next_code:n`. This function is documented on page 205.)

```

\_\_hook_clear_next:n
2102 <latexrelease> \IncludeInRelease{2023/06/01}{\_\_hook_clear_next:n}
2103 <latexrelease> {Hooks~with~args}
2104 \cs_new_protected:Npn \_\_hook_clear_next:n #1
2105   { \_\_hook_next_gset:nn {#1} { } }
2106 <latexrelease> \EndIncludeInRelease
2107 <latexrelease> \IncludeInRelease{2020/10/01}{\_\_hook_clear_next:n}
2108 <latexrelease> {Hooks~with~args}
2109 <latexrelease> \cs_gset_protected:Npn \_\_hook_clear_next:n #1
2110 <latexrelease> { \cs_gset_eq:cN { \_\_hook_next~#1 } \c_empty_tl }
2111 <latexrelease> \EndIncludeInRelease

(End of definition for \_\_hook_clear_next:n.)

```

4.9 Using the hook

`\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.

```

2112 <latexrelease> \IncludeInRelease{2023/06/01}{\hook_use:n}
2113 <latexrelease> {Hooks~with~args}
2114 \cs_new_protected:Npn \hook_use:n #1
2115   {
2116     \_\_hook_preamble_hook:n {#1}
2117     \_\_hook_use_initialized:n {#1}
2118   }
2119 \cs_new:Npn \_\_hook_use_initialized:n #1
2120   {
2121     \if_cs_exist:w \_\_hook~#1 \cs_end:
2122       \cs:w \_\_hook~#1 \use_i:nn
2123     \fi:
2124     \use_none:n
2125     \cs_end:
2126   }
2127 \cs_new_protected:Npn \_\_hook_preamble_hook:n #1
2128   {
2129     \if_cs_exist:w \_\_hook~#1 \cs_end:
2130       \_\_hook_initialize_hook_code:n {#1}
2131     \fi:

```

```

2132   }
2133   ⟨latexrelease⟩\EndIncludeInRelease
2134   ⟨latexrelease⟩\IncludeInRelease{2021/11/15}{\hook_use:n}
2135   ⟨latexrelease⟩                               {Standardise-generic-hook-names}
2136   ⟨latexrelease⟩\cs_new_protected:Npn \hook_use:n #1
2137   ⟨latexrelease⟩  {
2138     ⟨latexrelease⟩    \tl_if_exist:cT { __hook~#1 }
2139     ⟨latexrelease⟩    {
2140       ⟨latexrelease⟩      \__hook_preamble_hook:n {#1}
2141       ⟨latexrelease⟩      \cs:w __hook~#1 \cs_end:
2142     }
2143   }
2144   ⟨latexrelease⟩\cs_new:Npn \__hook_use_initialized:n #1
2145   ⟨latexrelease⟩  {
2146     ⟨latexrelease⟩    \if_cs_exist:w __hook~#1 \cs_end:
2147     ⟨latexrelease⟩    \cs:w __hook~#1 \exp_after:wN \cs_end:
2148     ⟨latexrelease⟩    \fi:
2149   }
2150   ⟨latexrelease⟩\cs_new_protected:Npn \__hook_preamble_hook:n #1
2151   ⟨latexrelease⟩  { \__hook_initialize_hook_code:n {#1} }
2152   ⟨latexrelease⟩\cs_new:Npn \hook_use:nw #1 { }
2153   ⟨latexrelease⟩\EndIncludeInRelease
2154   ⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\hook_use:n}
2155   ⟨latexrelease⟩                               {Standardise-generic-hook-names}
2156   ⟨latexrelease⟩\cs_new_protected:Npn \hook_use:n #1
2157   ⟨latexrelease⟩  {
2158     ⟨latexrelease⟩    \tl_if_exist:cTF { __hook~#1 }
2159     ⟨latexrelease⟩    {
2160       ⟨latexrelease⟩      \__hook_preamble_hook:n {#1}
2161       ⟨latexrelease⟩      \cs:w __hook~#1 \cs_end:
2162     }
2163     ⟨latexrelease⟩    { \__hook_use:wn #1 / \s__hook_mark {#1} }
2164   }
2165   ⟨latexrelease⟩\cs_new:Npn \__hook_use_initialized:n #1
2166   ⟨latexrelease⟩  {
2167     ⟨latexrelease⟩    \if_cs_exist:w __hook~#1 \cs_end:
2168     ⟨latexrelease⟩    \else:
2169     ⟨latexrelease⟩      \__hook_use undefined:w
2170     ⟨latexrelease⟩      \fi:
2171     ⟨latexrelease⟩      \cs:w __hook~#1 \__hook_use_end:
2172   }
2173   ⟨latexrelease⟩\cs_new:Npn \__hook_use undefined:w #1 #2 __hook~#3 \__hook_use_end:
2174   ⟨latexrelease⟩  {
2175     ⟨latexrelease⟩    #1 % fi
2176     ⟨latexrelease⟩    \__hook_use:wn #3 / \s__hook_mark {#3}
2177   }
2178   ⟨latexrelease⟩\cs_new_protected:Npn \__hook_preamble_hook:n #1
2179   ⟨latexrelease⟩  { \__hook_initialize_hook_code:n {#1} }
2180   ⟨latexrelease⟩\cs_new_eq:NN \__hook_use_end: \cs_end:
2181   ⟨latexrelease⟩\cs_new:Npn \hook_use:nw #1 { }
2182   ⟨latexrelease⟩\EndIncludeInRelease

```

(End of definition for `\hook_use:n`, `__hook_use_initialized:n`, and `__hook_preamble_hook:n`. This function is documented on page 204.)

```

\hook_use:nnw
\__hook_use_initialized:n nw 2183 { latexrelease } \IncludeInRelease{2023/06/01}{\hook_use:nnw}
2184 { latexrelease } { Hooks~with~args }
2185 \cs_new_protected:Npn \hook_use:nnw #1
2186 {
2187     \__hook_preamble_hook:n {#1}
2188     \__hook_use_initialized:n nw {#1}
2189 }
2190 \cs_new:Npn \__hook_use_initialized:n nw #1 #2
2191 {
2192     \cs:w
2193         \if_cs_exist:w __hook~#1 \cs_end:
2194             __hook~#1
2195         \else:
2196             use_none: \prg_replicate:nn {#2} { n }
2197         \fi:
2198     \cs_end:
2199 }
2200 { latexrelease } \EndIncludeInRelease
2201 { latexrelease } \IncludeInRelease{2020/10/01}{\hook_use:nnw}
2202 { latexrelease } { Hooks~with~args }
2203 { latexrelease } \cs_gset:Npn \hook_use:nnw #1 #2
2204 { latexrelease } { \use:c { use_none: \prg_replicate:nn {#2} { n } } }
2205 { latexrelease } \EndIncludeInRelease

```

(End of definition for `\hook_use:nnw` and `__hook_use_initialized:n nw`. This function is documented on page [204](#).)

```

\__hook_post_initialization_defs:
2206 { latexrelease } \IncludeInRelease{2023/06/01}{\__hook_post_initialization_defs:}
2207 { latexrelease } { Hooks~with~args }
2208 \cs_new_protected:Npn \__hook_post_initialization_defs:
2209 {
2210     \cs_gset_eq:NN \hook_use:n \__hook_use_initialized:n
2211     \cs_gset_eq:NN \hook_use:nnw \__hook_use_initialized:n nw
2212     \cs_gset_eq:NN \__hook_preamble_hook:n \use_none:n
2213     \cs_gset_eq:NN \__hook_post_initialization_defs: \prg_do_nothing:
2214 }
2215 { latexrelease } \EndIncludeInRelease
2216 { latexrelease } \IncludeInRelease{2020/10/01}{\__hook_post_initialization_defs:}
2217 { latexrelease } { Hooks~with~args }
2218 { latexrelease } \cs_undefine:N \__hook_post_initialization_defs:
2219 { latexrelease } \EndIncludeInRelease

```

(End of definition for `__hook_post_initialization_defs:.`)

`__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 exists.

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 exists, and calls `__hook_preamble_hook:n` if so, then uses the hook.

```

2220 <|latexrelease>\IncludeInRelease{2021/11/15}{\_\_hook\_use:wn}
2221 <|latexrelease>                                {Standardise-generic~hook~names}
2222 <|latexrelease>\EndIncludeInRelease
2223 <|latexrelease>\IncludeInRelease{2020/10/01}{\_\_hook\_use:wn}
2224 <|latexrelease>                                {Standardise-generic~hook~names}
2225 <|latexrelease>\cs_new:Npn \_\_hook\_use:wn #1 / #2 \s\_hook\_mark #3
2226 <|latexrelease>  {
2227 <|latexrelease>    \str_if_eq:nnTF {#1} { file } {
2228 <|latexrelease>      { \_\_hook_try_file_hook:n {#3} }
2229 <|latexrelease>      { } % Hook doesn't exist
2230 <|latexrelease>  }

2231 <|latexrelease>\cs_new_protected:Npn \_\_hook_try_file_hook:n #1
2232 <|latexrelease>  {
2233 <|latexrelease>    \_\_hook_if_file_hook:wTF #1 / \s\_hook_mark
2234 <|latexrelease>    {
2235 <|latexrelease>      \exp_args:Ne \_\_hook_if_usable_use:n
2236 <|latexrelease>      { \exp_args:Ne \_\_hook_file_hook_normalize:n {#1} }
2237 <|latexrelease>    }
2238 <|latexrelease>    { \_\_hook_if_usable_use:n {#1} } % file/ generic hook (e.g. file/before)
2239 <|latexrelease>  }

2240 <|latexrelease>\cs_new_protected:Npn \_\_hook_if_usable_use:n #1
2241 <|latexrelease>  {
2242 <|latexrelease>    \tl_if_exist:cT { __hook~#1 }
2243 <|latexrelease>    {
2244 <|latexrelease>      \_\_hook_preamble_hook:n {#1}
2245 <|latexrelease>      \cs:w __hook~#1 \cs_end:
2246 <|latexrelease>    }
2247 <|latexrelease>  }

2248 <|latexrelease>\EndIncludeInRelease

```

(End of definition for __hook_use:wn, __hook_try_file_hook:n, and __hook_if_usable_use:n.)

\hook_use_once:n
\hook_use_once:nnw

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.

```

2249 <|latexrelease>\IncludeInRelease{2023/06/01}{\hook_use_once:nnw}
2250 <|latexrelease>                                {Hooks~with~args}
2251 \cs_new_protected:Npn \hook_use_once:n #1
2252  {
2253   \_\_hook_if_execute_immediately:nF {#1}
2254   { \_\_hook_normalize_hook_args:Nn \_\_hook_use_once:nn { \use:n {#1} } { 0 } }
2255 }
2256 \cs_new_protected:Npn \hook_use_once:nnw #1 #2
2257  {
2258   \_\_hook_if_execute_immediately:nF {#1}
2259   { \_\_hook_normalize_hook_args:Nn \_\_hook_use_once:nn { \use:n {#1} } {#2 } }
2260 }
2261 <|latexrelease>\EndIncludeInRelease

```

(End of definition for `\hook_use_once:n` and `\hook_use_once:nnw`. These functions are documented on page 204.)

```
2262 <|latexrelease>\IncludeInRelease{2020/10/01}{\hook_use_once:nnw}
2263 <|latexrelease>                                {Hooks~with~args}
2264 <|latexrelease>\cs_gset_protected:Npn \hook_use_once:n #1
2265 <|latexrelease>  {
2266 <|latexrelease>    \__hook_if_execute_immediately:nF {#1}
2267 <|latexrelease>    { \__hook_normalize_hook_args:Nn \__hook_use_once:n { \use:n {#1} } }
2268 <|latexrelease>  }
2269 <|latexrelease>\cs_gset:Npn \hook_use_once:nnw #1 #2
2270 <|latexrelease>  { \use:c { use_none: \prg_replicate:nn {#2} { n } } }
2271 <|latexrelease>\EndIncludeInRelease
```

`__hook_use_once:nn`

```
2272 <|latexrelease>\IncludeInRelease{2023/06/01}{\__hook_use_once:nn}
2273 <|latexrelease>                                {Hooks~with~args}
2274 \cs_new_protected:Npn \__hook_use_once:nn #1 #2
2275  {
2276    \__hook_preamble_hook:n {#1}
2277    \__hook_use_once_set:n {#1}
```

When a hook has arguments, the call to `__hook_use_initialized:n`, should be the very last thing to happen, otherwise the arguments grabbed will be wrong. So, to clean up after the hook we need to cheat a bit and sneak the cleanup code at the end of the hook, along with the next execution code.

```
2278 \__hook_replacing_args_false:
2279 \__hook_cs_gput_right:nnn { _next } {#1} { \__hook_use_once_clear:n {#1} }
2280 \__hook_replacing_args_reset:
2281 \__hook_if_usable:nTF {#1}
2282   { \__hook_use_initialized:n {#1} }
2283   {
2284     \int_compare:nNnT {#2} > { 0 }
2285     { \use:c { use_none: \prg_replicate:nn {#2} { n } } }
2286   }
2287 }
2288 <|latexrelease>\EndIncludeInRelease
2289 %
2290 <|latexrelease>\IncludeInRelease{2020/10/01}{\__hook_use_once:nn}
2291 <|latexrelease>                                {Hooks~with~args}
2292 <|latexrelease>\cs_gset_protected:Npn \__hook_use_once:n #1
2293 <|latexrelease>  {
2294 <|latexrelease>    \__hook_preamble_hook:n {#1}
2295 <|latexrelease>    \__hook_use_once_set:n {#1}
2296 <|latexrelease>    \__hook_use_initialized:n {#1}
2297 <|latexrelease>    \__hook_use_once_clear:n {#1}
2298 <|latexrelease>  }
2299 <|latexrelease>\cs_undefine:N \__hook_use_once:nn
2300 <|latexrelease>\EndIncludeInRelease
```

(End of definition for `__hook_use_once:nn`.)

`__hook_use_once_set:n` `__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.

```

2301 <latexrelease>\IncludeInRelease{2023/06/01}{\_\_hook_use_once_clear:n}
2302 <latexrelease>                                {Hooks-with-args}
2303 \cs_new_protected:Npn \_\_hook_use_once_set:n #1
2304   { \_\_hook_tl_gset:cn { g__hook_#1_reversed_tl } { I } }
2305 \cs_new_protected:Npn \_\_hook_use_once_clear:n #1
2306   {
2307     \_\_hook_code_gset:nn {#1} { }
2308     \_\_hook_next_gset:nn {#1} { }
2309     \_\_hook_toplevel_gset:nn {#1} { }
2310     \prop_gclear_new:c { g__hook_#1_code_prop }
2311   }
2312 <latexrelease>\EndIncludeInRelease
2313 <latexrelease>\IncludeInRelease{2020/10/01}{\_\_hook_use_once_clear:n}
2314 <latexrelease>                                {Hooks-with-args}
2315 <latexrelease>\cs_new_protected:Npn \_\_hook_use_once_clear:n #1
2316 <latexrelease>  {
2317 <latexrelease>    \_\_hook_tl_gclear:c { __hook~#1 }
2318 <latexrelease>    \_\_hook_tl_gclear:c { __hook_next~#1 }
2319 <latexrelease>    \_\_hook_tl_gclear:c { __hook_toplevel~#1 }
2320 <latexrelease>    \prop_gclear_new:c { g__hook_#1_code_prop }
2321 <latexrelease>  }
2322 <latexrelease>\EndIncludeInRelease

```

(End of definition for __hook_use_once_set:n and __hook_use_once_clear:n.)

__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 \g__hook_{hook}_reversed_tl is I. The gymnastics around \if:w is there to allow the reversed token list to be empty.

```

2323 \prg_new_conditional:Npnn \_\_hook_if_execute_immediately:n #1 { T, F, TF }
2324   {
2325     \exp_after:wN \_\_hook_use_none_delimit_by_s_mark:w
2326     \if:w I
2327       \if_cs_exist:w g__hook_#1_reversed_tl \cs_end:
2328         \cs:w g__hook_#1_reversed_tl \exp_after:wN \cs_end:
2329       \fi:
2330       X
2331       \s__hook_mark \prg_return_true:
2332     \else:
2333       \s__hook_mark \prg_return_false:
2334     \fi:
2335   }

```

(End of 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.

```

2336 <latexrelease>\IncludeInRelease{2023/06/01}{\hook_if_empty:n}
2337 <latexrelease>                                {Hooks~with~args}
2338 \prg_new_conditional:Npnn \hook_if_empty:n #1 { p , T , F , TF }
2339 {
2340     \if:w
2341         T
2342         \prop_if_exist:cT { g_hook_#1_code_prop }
2343             { \prop_if_empty:cF { g_hook_#1_code_prop } { F } }
2344             \_hook_cs_if_empty:cF { __hook_toplevel~#1 } { F }
2345             \_hook_cs_if_empty:cF { __hook_next~#1 } { F }
2346         T
2347         \prg_return_true:
2348     \else:
2349         \prg_return_false:
2350     \fi:
2351 }
2352 <latexrelease>\EndIncludeInRelease
2353 <latexrelease>\IncludeInRelease{2020/10/01}{\hook_if_empty:n}
2354 <latexrelease>                                {Hooks~with~args}
2355 <latexrelease>\prg_new_conditional:Npnn \hook_if_empty:n #1 { p , T , F , TF }
2356 <latexrelease> {
2357 <latexrelease>     \_hook_if_structure_exist:nTF {#1}
2358 <latexrelease>     {
2359 <latexrelease>         \bool_lazy_and:nnTF
2360 <latexrelease>             { \prop_if_empty_p:c { g_hook_#1_code_prop } }
2361 <latexrelease>             {
2362 <latexrelease>                 \bool_lazy_and_p:nn
2363 <latexrelease>                     { \tl_if_empty_p:c { __hook_toplevel~#1 } }
2364 <latexrelease>                     { \tl_if_empty_p:c { __hook_next~#1 } }
2365 <latexrelease>                 }
2366 <latexrelease>             { \prg_return_true: }
2367 <latexrelease>             { \prg_return_false: }
2368 <latexrelease>         }
2369 <latexrelease>         { \prg_return_true: }
2370 <latexrelease>     }
2371 <latexrelease>\EndIncludeInRelease

```

(End of definition for `\hook_if_empty:nTF`. This function is documented on page 205.)

`__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).

```
2372 \prg_new_conditional:Npnn \__hook_if_usable:n #1 { p , T , F , TF }
2373 {
2374     \cs_if_exist:cTF { __hook~#1 }
2375     { \prg_return_true: }
2376     { \prg_return_false: }
2377 }
```

(End of 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`.

```
2378 \prg_new_conditional:Npnn \__hook_if_structure_exist:n #1 { p , T , F , TF }
2379 {
2380     \prop_if_exist:cTF { g__hook_#1_code_prop }
2381     { \prg_return_true: }
2382     { \prg_return_false: }
2383 }
```

(End of definition for `__hook_if_structure_exist:nTF`.)

`__hook_if_declared_p:n` Internal test to check if the hook was officially declared with `\hook_new:n` or a variant.
`__hook_if_declared:nTF`

```
2384 \prg_new_conditional:Npnn \__hook_if_declared:n #1 { p , T , F , TF }
2385 {
2386     \tl_if_exist:cTF { g__hook_#1_declared_tl }
2387     { \prg_return_true: }
2388     { \prg_return_false: }
2389 }
```

(End of definition for `__hook_if_declared:nTF`.)

`__hook_if_reversed_p:n` An internal conditional that checks if a hook is reversed.

`__hook_if_reversed:nTF`

```
2390 \prg_new_conditional:Npnn \__hook_if_reversed:n #1 { p , T , F , TF }
2391 {
2392     \exp_after:wN \__hook_use_none_delimit_by_s_mark:w
2393     \if:w - \cs:w g__hook_#1_reversed_tl \cs_end:
2394         \s__hook_mark \prg_return_true:
2395     \else:
2396         \s__hook_mark \prg_return_false:
2397     \fi:
2398 }
```

(End of 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.

`__hook_if_generic:nTF`

```
2399 \prg_new_conditional:Npnn \__hook_if_generic:n #1 { T , TF }
2400     { \__hook_if_generic:w #1 / / \s__hook_mark }
2401 \cs_new:Npn \__hook_if_generic:w #1 / #2 / #3 / #4 \s__hook_mark
```

```

2402  {
2403    \cs_if_exist:cTF { c__hook_generic_#1./#3_tl }
2404    { \prg_return_true: }
2405    { \prg_return_false: }
2406  }
2407 \prg_new_conditional:Npnn \__hook_if_deprecated_generic:n #1 { T, TF }
2408   { \__hook_if_deprecated_generic:w #1 / / / \s__hook_mark }
2409 \cs_new:Npn \__hook_if_deprecated_generic:w #1 / #2 / #3 / #4 \s__hook_mark
2410   {
2411     \cs_if_exist:cTF { c__hook_deprecated_#1./#2_tl }
2412     {
2413       \tl_if_empty:nTF {#3}
2414       { \prg_return_false: }
2415       { \prg_return_true: }
2416     }
2417     { \prg_return_false: }
2418   }

```

(End of definition for `__hook_if_generic:nTF` and `__hook_if_deprecated_generic:nTF`.)

`__hook_if_cmd_hook_p:n` An internal conditional that checks if a given hook is a valid generic cmd hook.

```

\__hook_if_cmd_hook:nTF 2419 \IncludeInRelease{2023/06/01}{\__hook_if_cmd_hook:n}
\__hook_if_cmd_hook:p:w 2420 \IncludeInRelease{}{Hooks-with-args}
\__hook_if_cmd_hook:wTF 2421 \prg_new_conditional:Npnn \__hook_if_cmd_hook:n #1 { T }
2422   { \__hook_if_cmd_hook:w #1 / / / \s__hook_mark }
2423 \cs_new:Npn \__hook_if_cmd_hook:w #1 / #2 / #3 / #4 \s__hook_mark
2424   {
2425     \if:w Y
2426       \str_if_eq:nnF {#1} { cmd } { N }
2427       \tl_if_exist:cF { c__hook_generic_#1./#3_tl } { N }
2428       Y
2429       \prg_return_true:
2430     \else:
2431       \prg_return_false:
2432     \fi:
2433   }
2434 \EndIncludeInRelease
2435 \IncludeInRelease{2020/10/01}{\__hook_if_cmd_hook:n}
2436 \IncludeInRelease{}{Hooks-with-args}
2437 \cs_undefine:N \__hook_if_cmd_hook:nT
2438 \EndIncludeInRelease

```

(End of definition for `__hook_if_cmd_hook:nTF` and `__hook_if_cmd_hook:wTF`.)

`__hook_if_generic_reversed_p:n` An internal conditional that checks if a name belongs to a generic reversed hook.

```

\__hook_if_generic_reversed:nTF 2439 \prg_new_conditional:Npnn \__hook_if_generic_reversed:n #1 { T }
2440   { \__hook_if_generic_reversed:w #1 / / / \scan_stop: }
2441 \cs_new:Npn \__hook_if_generic_reversed:w #1 / #2 / #3 / #4 \scan_stop:
2442   {
2443     \if_charcode:w - \cs:w c__hook_generic_#1./#3_tl \cs_end:
2444     \prg_return_true:
2445   \else:
2446     \prg_return_false:
2447   \fi:
2448 }

```

(End of definition for `_hook_if_generic_reversed:nTF`.)

```
\_hook_if_replacing_args:TF
  \_hook_misused_if_replacing_args:nn
\_\_hook_replacing_args_true:
  \_hook_replacing_args_false:
  \_hook_replacing_args_reset:
\g\_hook_replacing_stack_seq
2449 \seq_new:N \g\_hook_replacing_stack_seq
2450 \cs_new:Npn \_hook_misused_if_replacing_args:nn #1 #2
2451 {
  \msg_expandable_error:nnn { latex2e } { should-not-happen }
  { Misused~\_hook_if_replacing_args:.. }
}
2454 \cs_new:Npn \_hook_if_replacing_args:TF
2455 { \_hook_misused_if_replacing_args:nn }
2457 \cs_new_protected:Npn \_hook_replacing_args_true:
2458 {
  \seq_gpush:No \g\_hook_replacing_stack_seq
  { \_hook_if_replacing_args:TF }
  \cs_set:Npn \_hook_if_replacing_args:TF { \use_i:nn }
}
2462 }
2463 \cs_new_protected:Npn \_hook_replacing_args_false:
2464 {
  \seq_gpush:No \g\_hook_replacing_stack_seq
  { \_hook_if_replacing_args:TF }
  \cs_set:Npn \_hook_if_replacing_args:TF { \use_ii:nn }
}
2468 }
2469 \cs_new_protected:Npn \_hook_replacing_args_reset:
2470 {
  \seq_gpop:NN \g\_hook_replacing_stack_seq \l\_hook_return_tl
  \cs_gset_eq:NN \_hook_if_replacing_args:TF \l\_hook_return_tl
}
2473 }
```

(End of definition for `_hook_if_replacing_args:TF` and others.)

4.11 Messages

Hook errors are LaTeX kernel errors:

```
2474 \prop_gput:Nnn \g_msg_module_type_prop { hooks } { LaTeX }
And so are kernel errors (this should move elsewhere eventually).
2475 \prop_gput:Nnn \g_msg_module_type_prop { latex2e } { LaTeX }
2476 \prop_gput:Nnn \g_msg_module_name_prop { latex2e } { kernel }
2477 \msg_new:nnnn { hooks } { labels-incompatible }
2478 {
  Labels~'#1'~and~'#2'~are~incompatible
  \str_if_eq:nnF {#3} {??} { ~in~hook~'#3' } .~
  \int_compare:nNnTF {#4} = { 1 }
  { The~ code~ for~ both~ labels~ will~ be~ dropped. }
  { You~ may~ see~ errors~ later. }
}
2484 }
2485 { LaTeX-found~two~incompatible~labels~in~the~same~hook.~
2486   This~indicates~an~incompatibility~between~packages.~}
2487 \msg_new:nnnn { hooks } { exists }
2488 { Hook~'#1'~ has~ already~ been~ declared. }
2489 { There~ already~ exists~ a~ hook~ declaration~ with~ this~
```

```

2490      name.\\
2491      Please~ use~ a~ different~ name~ for~ your~ hook.}
2492  <latexrelease>\IncludeInRelease{2023/06/01}{too-many-args}
2493  <latexrelease>                      {Hooks~with~args}

2494 \msg_new:nnn { hooks } { too-many-args }
2495   { Too-many~arguments~for~hook~'#1'. }
2496   {
2497     You~tried~to~declare~a~hook~with~#2~arguments,~but~a~
2498     hook~can~only~have~up~to~nine.~LaTeX~will~define~this~
2499     hook~with~nine~arguments.
2500   }

2501 \msg_new:nnn { hooks } { without-args }
2502   { Hook~'#1'~has~no~arguments. }
2503   {
2504     You~tried~to~use~\iow_char:N\#2WithArguments~
2505     on~a~hook~that~takes~no~arguments.\\
2506     Check~the~usage~of~the~hook~or~use~\iow_char:N\#2~instead.\\
2507     \\
2508     LaTeX~will~use~\iow_char:N\#2.
2509   }

2510 \msg_new:nnn { hooks } { one-time-args }
2511   { You~can't~have~arguments~in~used~one-time~hook~'#1'. }
2512   {
2513     You~tried~to~use~\iow_char:N\#2WithArguments~
2514     on~a~one-time~hook~that~has~already~been~used.~
2515     You~have~to~add~the~code~before~the~hook~is~used,~
2516     or~add~the~code~without~arguments~using~\iow_char:N\#2~instead.\\
2517     \\
2518     LaTeX~will~use~\iow_char:N\#2.
2519   }

2520 <latexrelease>\EndIncludeInRelease
2521 <latexrelease>\IncludeInRelease{2020/10/01}{too-many-args}
2522 <latexrelease>                      {Hooks~with~args}
2523 <latexrelease>\EndIncludeInRelease

2524 \msg_new:nnn { hooks } { hook-disabled }
2525   { Cannot~add~code~to~disabled~hook~'#1'. }
2526   {
2527     The~hook~'#1'~you~tried~to~add~code~to~was~previously~disabled~
2528     with~\iow_char:N\hook_disable_generic:n~or~\iow_char:N\DisableGenericHook,~so~
2529     it~cannot~have~code~added~to~it.
2530   }

2531 \msg_new:nnn { hooks } { empty-label }
2532   {
2533     Empty~code~label~\msg_line_context:~.
2534     Using~'\_hook_curname_or_default:'~instead.
2535   }

2536 \msg_new:nnn { hooks } { no-default-label }
2537   {
2538     Missing~(empty)~default~label~\msg_line_context:~\\
2539     This~command~was~ignored.
2540   }

```

```

2541 \msg_new:nnn { hooks } { unknown-rule }
2542 {
2543     Unknown~ relationship~ '#3'~
2544     between~ labels~ '#2'~ and~ '#4'~
2545     \str_if_eq:nnF {#1} {??} { ~in~hook~'#1' }. ~
2546     Perhaps~ a~ misspelling?
2547 }
2548 {
2549     The~ relation~ used~ not~ known~ to~ the~ system.~ Allowed~ values~ are~
2550     'before'~ or~ '<',~
2551     'after'~ or~ '>',~
2552     'incompatible-warning',~
2553     'incompatible-error',~
2554     'voids'~ or~
2555     'unrelated'.
2556 }
2557 \msg_new:nnn { hooks } { rule-too-late }
2558 {
2559     Sorting~rule~for~'#1'~hook~applied~too~late.\\
2560     Try~setting~this~rule~earlier.
2561 }
2562 {
2563     You~tried~to~set~the~ordering~of~hook~'#1'~using\\
2564     \\ \iow_char:N \\DeclareHookRule{#1}{#2}{#3}{#4}\\
2565     but~hook~'#1'~was~already~used~as~a~one-time~hook,~
2566     thus~sorting~is\\
2567     no~longer~possible.~Declare~the~rule~
2568     before~the~hook~is~used.
2569 }
2570 \msg_new:nnn { hooks } { misused-top-level }
2571 {
2572     Illegal~use~of~\iow_char:N \\AddToHook{#1}[top-level]{...}.\\
2573     'top-level'~is~reserved~for~the~user's~document.
2574 }
2575 {
2576     The~'top-level'~label~is~meant~for~user~code~only,~and~should~only~
2577     be~used~(sparingly)~in~the~main~document.~Use~the~default~label~
2578     '\_hook_currname_or_default:'~for~this~\@cls@pkg,~or~another~
2579     suitable~label.
2580 }
2581 \msg_new:nnn { hooks } { set-top-level }
2582 {
2583     You~cannot~change~the~default~label~#1~'top-level'.~Illegal \\
2584     \use:nn { ~ } { ~ } \iow_char:N \\#2{#3} \\
2585     \msg_line_context:.
2586 }
2587 \msg_new:nnn { hooks } { extra-pop-label }
2588 {
2589     Extra~\iow_char:N \\PopDefaultHookLabel. \\
2590     This~command~will~be~ignored.
2591 }
2592 \msg_new:nnn { hooks } { missing-pop-label }
2593 {

```

```

2594     Missing-\iow_char:N \\PopDefaultHookLabel. \\
2595     The-label-'#1'-was-pushed-but-never-popped.-Something-is-wrong.
2596 }
2597 \msg_new:nnn { latex2e } { should-not-happen }
2598 {
2599     This-should-not-happen.-#1 \\
2600     Please-report-at-https://github.com/latex3/latex2e.
2601 }
2602 \msg_new:nnn { hooks } { activate-disabled }
2603 {
2604     Cannot- activate- hook- '#1'- because- it- is- disabled!
2605 }
2606 \msg_new:nnn { hooks } { cannot-remove }
2607 {
2608     Cannot-remove-chunk-'#2'-from-hook-'#1'-because-
2609     \_\_hook_if_structure_exist:nTF {#1}
2610     { it-does-not-exist-in-that-hook. }
2611     { the-hook-does-not-exist. }
2612 }
2613 \msg_new:nnn { hooks } { generic-deprecated }
2614 {
2615     Generic-hook-'#1/#2/#3'-is-deprecated. \\
2616     Use-hook-'#1/#3/#2'-instead.
2617 }

```

4.12 L^AT_EX 2 _{ϵ} package interface commands

\NewHook
 \NewReversedHook
 \NewMirroredHookPair

```

2618 \NewDocumentCommand \NewHook           { m }
2619   { \hook_new:n {#1} }
2620 \NewDocumentCommand \NewReversedHook   { m }
2621   { \hook_new_reversed:n {#1} }
2622 \NewDocumentCommand \NewMirroredHookPair { mm }
2623   { \hook_new_pair:nn {#1}{#2} }

(End of definition for \NewHook, \NewReversedHook, and \NewMirroredHookPair. These functions are
documented on page 192.)
```

\NewHookWithArguments
 \NewReversedHookWithArguments
 \NewMirroredHookPairWithArguments

Declaring new hooks with arguments...

```

2624 \IfFileExists{2023/06/01}{\NewHookWithArguments}
2625 \IfFileExists{}{\NewReversedHookWithArguments}
2626 \NewDocumentCommand \NewHookWithArguments      { mm }
2627   { \hook_new_with_args:nn {#1} {#2} }
2628 \NewDocumentCommand \NewReversedHookWithArguments { mm }
2629   { \hook_new_reversed_with_args:nn {#1} {#2} }
2630 \NewDocumentCommand \NewMirroredHookPairWithArguments { mmm }
2631   { \hook_new_pair_with_args:nnn {#1} {#2} {#3} }
2632 \IfFileExists{}{\EndIncludeInRelease}
2633 \IfFileExists{2020/10/01}{\NewHookWithArguments}
2634 \IfFileExists{}{\NewReversedHookWithArguments}
2635 \cs_new_protected:Npn \NewHookWithArguments #1 #2 { }
2636 \cs_new_protected:Npn \NewReversedHookWithArguments #1 #2 { }
```

```
2637 ⟨latexrelease⟩\cs_new_protected:Npn \NewMirroredHookPairWithArguments #1 #2 #3 { }
2638 ⟨latexrelease⟩\EndIncludeInRelease
```

(End of definition for `\NewHookWithArguments`, `\NewReversedHookWithArguments`, and
`\NewMirroredHookPairWithArguments`. These functions are documented on page 193.)

```
2639 ⟨latexrelease⟩\IncludeInRelease{2021/06/01}{\ActivateGenericHook}
2640 ⟨latexrelease⟩                                {Providing~hooks}
```

\ActivateGenericHook Providing new hooks ...

```
2641 \NewDocumentCommand \ActivateGenericHook { m }
2642   { \hook_activate_generic:n {#1} }
```

(End of definition for `\ActivateGenericHook`. This function is documented on page 193.)

\DisableGenericHook Disabling a generic hook.

```
2643 \NewDocumentCommand \DisableGenericHook { m }
2644   { \hook_disable_generic:n {#1} }
```

(End of definition for `\DisableGenericHook`. This function is documented on page 193.)

```
2645 ⟨latexrelease⟩\EndIncludeInRelease
```

```
2646 ⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\ActivateGenericHook}
2647 ⟨latexrelease⟩                                {Providing~hooks}
2648 ⟨latexrelease⟩\def \ActivateGenericHook #1 { }
2649 ⟨latexrelease⟩\def \DisableGenericHook #1 { }
2650 ⟨latexrelease⟩\EndIncludeInRelease
```

\AddToHook

\AddToHookWithArguments

```
2651 ⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\AddToHookWithArguments}
2652 ⟨latexrelease⟩                                {Hooks~with~args}
2653 \NewDocumentCommand \AddToHook { m o +m }
2654   { \hook_gput_code:nnn {#1} {#2} {#3} }
2655 \NewDocumentCommand \AddToHookWithArguments { m o +m }
2656   { \hook_gput_code_with_args:nnn {#1} {#2} {#3} }
2657 ⟨latexrelease⟩\EndIncludeInRelease
2658 ⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\AddToHookWithArguments}
2659 ⟨latexrelease⟩                                {Hooks~with~args}
2660 ⟨latexrelease⟩\cs_new_protected:Npn \AddToHookWithArguments #1 #2 #3 { }
2661 ⟨latexrelease⟩\EndIncludeInRelease
```

(End of definition for `\AddToHook` and `\AddToHookWithArguments`. These functions are documented on page 195.)

\AddToHookNext

\AddToHookNextWithArguments

```
2662 ⟨latexrelease⟩\IncludeInRelease{2023/06/01}{\AddToHookNextWithArguments}
2663 ⟨latexrelease⟩                                {Hooks~with~args}
2664 \NewDocumentCommand \AddToHookNext { m +m }
2665   { \hook_gput_next_code:nn {#1} {#2} }
2666 \NewDocumentCommand \AddToHookNextWithArguments { m +m }
2667   { \hook_gput_next_code_with_args:nn {#1} {#2} }
2668 ⟨latexrelease⟩\EndIncludeInRelease
2669 ⟨latexrelease⟩\IncludeInRelease{2020/10/01}{\AddToHookNextWithArguments}
2670 ⟨latexrelease⟩                                {Hooks~with~args}
2671 ⟨latexrelease⟩\cs_new_protected:Npn \AddToHookNextWithArguments #1 #2 { }
2672 ⟨latexrelease⟩\EndIncludeInRelease
```

(End of definition for `\AddToHookNext` and `\AddToHookNextWithArguments`. These functions are documented on page 196.)

`\ClearHookNext`

```
2673 \NewDocumentCommand \ClearHookNext { m }
2674   { \hook_gclear_next_code:n {#1} }
```

(End of definition for `\ClearHookNext`. This function is documented on page 197.)

`\RemoveFromHook`

```
2675 \NewDocumentCommand \RemoveFromHook { m o }
2676   { \hook_gremove_code:nn {#1} {#2} }
```

(End of definition for `\RemoveFromHook`. This function is documented on page 195.)

`\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`

```
2677 \NewDocumentCommand \SetDefaultHookLabel { m }
2678   { \__hook_set_default_hook_label:n {#1} }
2679 %
2680 %   The label is only automatically updated with \cs{@onefilewithoptions}
2681 %   (\cs{usepackage} and \cs{documentclass}), but some packages, like
2682 %   Ti\emph{k}Z, define package-like interfaces, like
2683 %   \cs{usetikzlibrary} that are wrappers around \cs{input}, so they
2684 %   inherit the default label currently in force (usually |top-level|,
2685 %   but it may change if loaded in another package). To provide a
2686 %   package-like behavior also for hooks in these files, we provide
2687 %   high-level access to the default label stack.
2688 %   \begin{macrocode}
2689 \NewDocumentCommand \PushDefaultHookLabel { m }
2690   { \__hook_curr_name_push:n {#1} }
2691 \NewDocumentCommand \PopDefaultHookLabel { }
2692   { \__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`.

```
2693 \cs_gset_protected:Npn \@expl@push@filename@aux@@ #1#2#3
2694   {
2695     \__hook_curr_name_push:n {#3}
2696     \str_gset:Nx \g_file_curr_name_str {#3}
2697     #1 #2 {#3}
2698   }
```

(End of definition for `\SetDefaultHookLabel`, `\PushDefaultHookLabel`, and `\PopDefaultHookLabel`. These functions are documented on page 199.)

`\UseHook`

Avoid the overhead of `xparse` and its protection that we don't want here (since the hook should vanish without trace if empty)!

`\UseOneTimeHook`

```
2699 (latexrelease)\IncludeInRelease{2023/06/01}{\UseHookWithArguments}
2700 (latexrelease)           {Hooks-with-args}
2701 \cs_new:Npn \UseHook      { \hook_use:n }
```

```

2702 \cs_new:Npn \UseOneTimeHook { \hook_use_once:n }
2703 \cs_new:Npn \UseHookWithArguments           { \hook_use:nnw }
2704 \cs_new:Npn \UseOneTimeHookWithArguments { \hook_use_once:nnw }
2705 \langle latexrelease \rangle \EndIncludeInRelease
2706 \langle latexrelease \rangle \IncludeInRelease{2020/10/01}{\UseHookWithArguments}
2707 \langle latexrelease \rangle                   {Hooks-with-args}
2708 \langle latexrelease \rangle \cs_new:Npn \UseHookWithArguments #1 #2 { }
2709 \langle latexrelease \rangle \cs_new:Npn \UseOneTimeHookWithArguments #1 #2 { }
2710 \langle latexrelease \rangle \EndIncludeInRelease

```

(End of definition for `\UseHook` and others. These functions are documented on page [194](#).)

`\ShowHook`

```

\LogHook 2711 \cs_new_protected:Npn \ShowHook { \hook_show:n }
2712 \cs_new_protected:Npn \LogHook { \hook_log:n }

```

(End of definition for `\ShowHook` and `\LogHook`. These functions are documented on page [202](#).)

`\DebugHooksOn`

`\DebugHooksOff`

```

2713 \cs_new_protected:Npn \DebugHooksOn { \hook_debug_on: }
2714 \cs_new_protected:Npn \DebugHooksOff { \hook_debug_off: }

```

(End of definition for `\DebugHooksOn` and `\DebugHooksOff`. These functions are documented on page [203](#).)

`\DeclareHookRule`

```

2715 \NewDocumentCommand \DeclareHookRule { m m m m }
2716           { \hook_gset_rule:nnnn {#1}{#2}{#3}{#4} }

```

(End of definition for `\DeclareHookRule`. This function is documented on page [200](#).)

`\DeclareDefaultHookRule`

This declaration is only supported before `\begin{document}`.

```

2717 \NewDocumentCommand \DeclareDefaultHookRule { m m m }
2718           { \hook_gset_rule:nnnn {??}{#1}{#2}{#3} }
2719 \onlypreamble\DeclareDefaultHookRule

```

(End of definition for `\DeclareDefaultHookRule`. This function is documented on page [201](#).)

`\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?

```

2720 \NewDocumentCommand \ClearHookRule { m m m }
2721           { \hook_gset_rule:nnnn {#1}{#2}{unrelated}{#3} }

```

(End of definition for `\ClearHookRule`. This function is documented on page [201](#).)

`\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.

```

2722 \cs_new_eq:NN \IfHookEmptyTF \hook_if_empty:nTF

```

(End of definition for `\IfHookEmptyTF`. This function is documented on page [201](#).)

`\IfHookExistsTF`

Marked for removal and no longer documented in the doc section!

PhO: \IfHookExistsTF is used in jlreq.cls, pxtbegshi.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.

2723 \cs_new_eq:NN \IfHookExistsTF __hook_if_usable:nTF

(End of definition for \IfHookExistsTF.)

4.13 Deprecated that needs cleanup at some point

\hook_disable:n Deprecated.

```

\hook_provide:n 2724 \cs_new_protected:Npn \hook_disable:n
\hook_provide_reversed:n 2725 {
\hook_provide_pair:nn 2726   \__hook_deprecated_warn:nn
\__hook_activate_generic_reversed:n 2727   { hook_disable:n }
\__hook_activate_generic_pair:nn 2728   { hook_disable_generic:n }
2729   \hook_disable_generic:n
2730 }
2731 \cs_new_protected:Npn \hook_provide:n
2732 {
2733   \__hook_deprecated_warn:nn
2734   { hook_provide:n }
2735   { hook_activate_generic:n }
2736   \hook_activate_generic:n
2737 }
2738 \cs_new_protected:Npn \hook_provide_reversed:n
2739 {
2740   \__hook_deprecated_warn:nn
2741   { hook_provide_reversed:n }
2742   { hook_activate_generic:n }
2743   \__hook_activate_generic_reversed:n
2744 }
2745 \cs_new_protected:Npn \hook_provide_pair:nn
2746 {
2747   \__hook_deprecated_warn:nn
2748   { hook_provide_pair:nn }
2749   { hook_activate_generic:n }
2750   \__hook_activate_generic_pair:nn
2751 }
2752 \cs_new_protected:Npn \__hook_activate_generic_reversed:n #1
2753 { \__hook_normalize_hook_args:Nn \__hook_activate_generic:nn {#1} { - } }
2754 \cs_new_protected:Npn \__hook_activate_generic_pair:nn #1#2
2755 { \hook_activate_generic:n {#1} \__hook_activate_generic_reversed:n {#2} }
```

(End of definition for \hook_disable:n and others.)

\DisableHook Deprecated.

```

\ProvideHook 2756 \cs_new_protected:Npn \DisableHook
\ProvideReversedHook 2757 {
\ProvideMirroredHookPair 2758   \__hook_deprecated_warn:nn
2759   { DisableHook }
2760   { DisableGenericHook }
2761   \hook_disable_generic:n
2762 }
2763 \cs_new_protected:Npn \ProvideHook
```

```

2764  {
2765      \__hook_deprecated_warn:nn
2766      { ProvideHook }
2767      { ActivateGenericHook }
2768      \hook_activate_generic:n
2769  }
2770 \cs_new_protected:Npn \ProvideReversedHook
2771  {
2772      \__hook_deprecated_warn:nn
2773      { ProvideReversedHook }
2774      { ActivateGenericHook }
2775      \__hook_activate_generic_reversed:n
2776  }
2777 \cs_new_protected:Npn \ProvideMirroredHookPair
2778  {
2779      \__hook_deprecated_warn:nn
2780      { ProvideMirroredHookPair }
2781      { ActivateGenericHook }
2782      \__hook_activate_generic_pair:nn
2783  }

```

(End of definition for `\DisableHook` and others.)

`__hook_DEPRECATED_WARN:NN` Warns about a deprecation, telling what should be used instead.

```

2784 \cs_new_protected:Npn \__hook_DEPRECATED_WARN:nn #1 #2
2785  { \msg_warning:nnnn { hooks } { deprecated } { #1 } { #2 } }
2786 \msg_new:nnn { hooks } { deprecated }
2787  {
2788      Command-\iow_char:N\#1-is-deprecated-and-will-be-removed-in-a-
2789      future-release. \\ \\
2790      Use-\iow_char:N\#2-instead.
2791  }

```

(End of 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_E 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.

```
2792 <@@=
```

```

\@expl@@@initialize@all@@
\@expl@@@hook@curr@name@pop@@
2793 \cs_new_eq:NN \@expl@@@initialize@all@@
2794     \__hook_initialize_all:
2795 \cs_new_eq:NN \@expl@@@hook@curr@name@pop@@
2796     \__hook_curr_name_pop:

```

(End of definition for `\@expl@@@initialize@all@@` and `\@expl@@@hook@curr@name@pop@@`.)

Rolling back here doesn’t undefine the interface commands as they may be used in packages without rollback functionality. So we just make them do nothing which may or may not work depending on the code usage.

```
2797 %
```

```

2798 <|latexrelease>\IncludeInRelease{0000/00/00}{lthooks}
2799 <|latexrelease>                                {The~hook~management}%
2800 <|latexrelease>
2801 <|latexrelease>\def \NewHook#1{}
2802 <|latexrelease>\def \NewReversedHook#1{}
2803 <|latexrelease>\def \NewMirroredHookPair#1#2{}
2804 <|latexrelease>
2805 <|latexrelease>\def \DisableGenericHook #1{}
2806 <|latexrelease>
2807 <|latexrelease>\long\def\AddToHookNext#1#2{}
2808 <|latexrelease>
2809 <|latexrelease>\def\AddToHook#1{\@gobble@AddToHook@args}
2810 <|latexrelease>\providecommand\@gobble@AddToHook@args[2]{}{}
2811 <|latexrelease>
2812 <|latexrelease>\def\RemoveFromHook#1{\@gobble@RemoveFromHook@arg}
2813 <|latexrelease>\providecommand\@gobble@RemoveFromHook@arg[1]{}{}
2814 <|latexrelease>
2815 <|latexrelease>\def \UseHook          #1{}
2816 <|latexrelease>\def \UseOneTimeHook #1{}
2817 <|latexrelease>\def \ShowHook #1{}
2818 <|latexrelease>\let \DebugHooksOn \empty
2819 <|latexrelease>\let \DebugHooksOff\empty
2820 <|latexrelease>
2821 <|latexrelease>\def \DeclareHookRule #1#2#3#4{}
2822 <|latexrelease>\def \DeclareDefaultHookRule #1#2#3{}
2823 <|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.

```

2824 <|latexrelease>\long\def \IfHookExistsTF #1#2#3{#3}
2825 <|latexrelease>\long\def \IfHookEmptyTF #1#2#3{#2}
2826 <|latexrelease>
2827 <|latexrelease>\EndModuleRelease
2828 <|@@=hook>

2829 <|latexrelease>\cs:w __hook_rollback_tidyng: \cs_end:
2830 <|latexrelease>\bool_lazy_and:nnt
2831 <|latexrelease>  { \int_compare_p:nNn { \sourceLaTeXdate } > { 20230600 } }
2832 <|latexrelease>  { \int_compare_p:nNn { \requestedLaTeXdate } < { 20230601 } }
2833 <|latexrelease>  {
2834 <|latexrelease>    \cs_gset_protected:Npn __hook_rollback_tidyng:
2835 <|latexrelease>    {
2836 <|latexrelease>      \Olatex@error { Rollback-code-executed-twice }
2837 <|latexrelease>      {
2838 <|latexrelease>        Something~went~wrong~(unless~this~was~
2839 <|latexrelease>          done~on~purpose~in~a~testing~environment).
2840 <|latexrelease>      }
2841 <|latexrelease>      \use_none:nnnn
2842 <|latexrelease>    }
2843 <|latexrelease>    \cs_set:Npn __hook_tmp:w #1 #2
2844 <|latexrelease>    {
2845 <|latexrelease>      __hook_tl_gset:cx { __hook#1~#2 }
2846 <|latexrelease>      {

```

```

2847 <|latexrelease>          \exp_args:No \exp_not:o
2848 <|latexrelease>          {
2849 <|latexrelease>          \cs:w __hook#1~#2 \exp_last_unbraced:Nc \cs_end:
2850 <|latexrelease>          { __hook_braced_cs_parameter:n { __hook#1~#2 } }
2851 <|latexrelease>          }
2852 <|latexrelease>          }
2853 <|latexrelease>          \seq_map_inline:Nn \g__hook_all_seq
2854 <|latexrelease>          {
2855 <|latexrelease>          \exp_after:wN \cs_gset_nopar:Npn
2856 <|latexrelease>          \cs:w g__hook_#1_code_prop \exp_args:NNo \exp_args:No
2857 <|latexrelease>          \cs_end: { \cs:w g__hook_#1_code_prop \cs_end: }
2858 <|latexrelease>          \__hook_tmp:w { _toplevel } {#1}
2859 <|latexrelease>          \__hook_tmp:w { _next } {#1}
2860 <|latexrelease>
2861 <|latexrelease>          }
2862 <|latexrelease>          }
2863 \ExplSyntaxOff
2864 </2ekernel | latexrelease>
2865 <@@=>

```

File i

ltcmdhooks.dtx

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`,¹² `\AddToHookNext`, `\AddToHookWithArguments`, and `\AddToHookNextWithArguments`.¹³ These are:

`cmd/\langle name \rangle/before` This hook is executed at the very start of the command, right after its arguments (if any) are parsed. The hook `\langle code \rangle` runs in the command inside a call to `\UseHookWithArguments`. Any code added to this hook using `\AddToHookWithArguments` or `\AddToHookNextWithArguments` can access the command's arguments using `#1`, `#2`, etc., up to the number of arguments of the command. If `\AddToHook` or `\AddToHookNext` are used, the arguments cannot be accessed (see the `lthooks` documentation¹⁴ on hooks with arguments).

`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}`¹⁵ (i.e., using a command in the preamble will never execute the hook) 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]{%
  \UseHookWithArguments{cmd/foo/before}{2}{#1}{#2}%
  Code #1 for #2!}
```

and similarly `\AddToHook{cmd/foo/after}{<after code>}` alters the definition to

```
\renewcommand\foo[2]{%
  Code #1 for #2!%
  \UseHookWithArguments{cmd/foo/after}{2}{#1}{#2}}
```

¹²In this documentation, when something is being said about `\AddToHook`, the same will be valid for `\AddToHookWithArguments`, unless that particular paragraph is highlighting the differences between both. The same is true for the other hook-related functions and their ...`WithArguments` counterparts.

¹³In 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.

¹⁴`texdoc lthooks-doc`

¹⁵More specifically, they are inserted in the commands after the `begindocument` hook, so they are also not present while L^AT_EX is reading the `.aux` file.

In other words, the mechanism is similar to what `etoolbox` offers with `\preto{cmd}` and `\appto{cmd}` 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.

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.¹⁶

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.

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 `\UseHookWithArguments` calls inside the command in the proper positions, and manually define the hooks with `\NewHookWithArguments` or `\NewReversedHookWithArguments`. 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{%
    \UseHookWithArguments{cmd/fancybox/before}{2}{#1}{#2}%
    \parbox{#1}{#2}%
    \UseHookWithArguments{cmd/fancybox/after}{2}{#1}{#2}}}
```

¹⁶We might change this behavior in the main document slightly after gaining some usage experience.

```
\NewHookWithArguments{cmd/fancybox/before}{2}
\NewReversedHookWithArguments{cmd/fancybox/after}{2}
```

then the hooks will be executed where they should and no patching will be attempted. It is important that the hooks are declared with `\NewHookWithArguments` or `\NewReversedHookWithArguments`, otherwise the command hook code will try to patch the command. Note also that the call to `\UseHookWithArguments{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 `\DisableGenericHook`¹⁷, so that when someone 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.

3.1 Arguments and redefining commands

The code in `ltcmdhooks` does its best to find out how many arguments a given command has, and to insert the appropriate call to `\UseHookWithArguments`, so that the arguments seen by the hook are exactly those grabbed by the command (the hook, after all, is a macro call, so the arguments have to be placed in the right order, or they won't match).

When using the package writer interface, as discussed in section 3, to change the position of the hooks in your commands, you are also free to change how the hook code in your command sees its arguments. When a `cmd` hook is declared with `\NewHook` (or `\NewHookWithArguments` or other variations of that), it loses its "generic" nature and works as a regular hook. This means that you may choose to declare it without arguments regardless if the command takes arguments or not, or declare it with arguments, even if the command takes none.

However, this flexibility should not be abused. When using a nonstandard configuration for the hook arguments, think reasonably: a user will expect that the argument `#1` in the hook corresponds to the argument's first argument, and so on. Any other configuration is likely to cause confusion and, if used, will have to be well documented.

This flexibility, however, allows you to "correct" the arguments for the hooks. For example, L^AT_EX's `\refstepcounter` has a single argument, the name of the counter. The `cleveref` package adds an optional argument to `\refstepcounter`, making the name of the counter argument `#2`. If the author of `cleveref` wanted, for whatever reason, to add hooks to `\refstepcounter`, to preserve compatibility he could write something along the lines of:

¹⁷Please use `\DisableGenericHook` if at all, only on hooks that you "own", i.e., for commands your package or class defines and not second guess whether or not hooks of other packages should get disabled!

```
\NewHookWithArguments{cmd/refstepcounter/before}{1}
\renewcommand\refstepcounter[2] [⟨default⟩]{%
  \UseHookWithArguments{cmd/refstepcounter/before}{1}{#2}%
  <code for \refstepcounter>}
```

so that the mandatory argument, which is arg #2 in the definition, would still be seen as #1 in the hook code.

Another possibility would be to place the optional argument as the second argument for the hook, so that people looking for it would be able to use it. In either case, it would have to be well documented to cause as little confusion as possible.

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 #₁₂¹² 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`¹⁸, `\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}`

¹⁸It'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 of 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 of 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 of definition for \l__hook_patch_prefixes_tl, \l__hook_param_text_tl, and \l__hook_replace_text_tl.)
\c__hook_hash_tl \c__hook_hashes_tl	Two constant token lists that contain one and two parameter tokens. 11 \tl_const:Nn \c__hook_hash_tl { # } 12 \tl_const:Nn \c__hook_hashes_tl { # # } (End of definition for \c__hook_hash_tl and \c__hook_hashes_tl.)
__hook_exp_not:NN __hook_def_cmd:w	Two temporary macros that change depending on the macro being patched. 13 \cs_new_eq:NN __hook_exp_not:NN ? 14 \cs_new_eq:NN __hook_def_cmd:w ? (End of 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. 15 \quark_new:N \q__hook_recursion_tail 16 \quark_new:N \q__hook_recursion_stop (End of 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. 17 \prop_new:N \g__hook_delayed_patches_prop (End of definition for \g__hook_delayed_patches_prop.)
__hook_patch_debug:x	A helper for patching debug info. 18 \cs_new_protected:Npn __hook_patch_debug:x #1 19 { __hook_debug:n { \iow_term:x { [lthooks]~#1 } } } (End of definition for __hook_patch_debug:x.)

4.3 Variants

```
\tl_rescan:nV \expl3 function variants used throughout the code.
20  \cs_generate_variant:Nn \tl_rescan:nn { nV }

(End of 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.

```
21  \langle latexrelease \rangle \IncludeInRelease{2021/11/15}{\_hook_try_put_cmd_hook:n}%
22  \langle latexrelease \rangle \StandardiseGenericHookNames
23  \cs_new_protected:Npn \_hook_try_put_cmd_hook:n #1
24  { \_hook_try_put_cmd_hook:w #1 // \s__hook_mark {#1} }
25  \cs_new_protected:Npn \_hook_try_put_cmd_hook:w
26  #1 / #2 / #3 / #4 \s__hook_mark #5
27  {
28  \_hook_debug:n { \iow_term:n { ->~Adding~cmd~hook~to~'#2'~(#3): } }
29  \exp_args:Nc \_hook_patch_cmd_or_delay:Nnn {#2} {#2} {#3}
30  }
31  \langle latexrelease \rangle \EndIncludeInRelease
32  \langle latexrelease \rangle \IncludeInRelease{2021/06/01}{\_hook_try_put_cmd_hook:n}%
33  \langle latexrelease \rangle \StandardiseGenericHookNames
34  \langle latexrelease \rangle \cs_new_protected:Npn \_hook_try_put_cmd_hook:n #1
35  \langle latexrelease \rangle { \_hook_try_put_cmd_hook:w #1 // \s__hook_mark {#1} }
36  \langle latexrelease \rangle \cs_new_protected:Npn \_hook_try_put_cmd_hook:w
37  #1 / #2 / #3 / #4 \s__hook_mark #5
38  \langle latexrelease \rangle {
39  \langle latexrelease \rangle \_hook_debug:n { \iow_term:n { ->~Adding~cmd~hook~to~'#2'~(#3): } }
40  \langle latexrelease \rangle \str_case:nnTF {#3}
41  { { before } { } { after } { } }
42  \langle latexrelease \rangle { \exp_args:Nc \_hook_patch_cmd_or_delay:Nnn {#2} {#2} {#3} }
43  \langle latexrelease \rangle { \msg_error:nnnn { hooks } { wrong-cmd-hook } {#2} {#3} }
44  \langle latexrelease \rangle }
45  \langle latexrelease \rangle \EndIncludeInRelease
```

(End of definition for `_hook_try_put_cmd_hook:n` and `_hook_try_put_cmd_hook:w`.)

```
\_hook_patch_cmd_or_delay:Nnn
\_hook_cmd_begin_document_code:
```

In the preamble, `_hook_patch_cmd_or_delay:Nnn` just adds the patch instruction to a property list to be executed later.

```
46  \cs_new_protected:Npn \_hook_patch_cmd_or_delay:Nnn #1 #2 #3
47  {
48  \_hook_debug:n { \iow_term:n { ->~Add~generic~cmd~hook~for~#2~(#3). } }
49  \_hook_debug:n
50  { \iow_term:n { !~In~the~preamble:~delaying. } }
51  \prop_gput:Nnn \g__hook_delayed_patches_prop { #2 / #3 }
52  { \_hook_cmd_try_patch:nn {#2} {#3} }
53 }
```

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.

```

54 \cs_new_protected:Npn \_\_hook_cmd_begindocument_code:
55 {
56   \cs_gset_eq:NN \_\_hook_patch_cmd_or_delay:Nnn \_\_hook_patch_command:Nnn
57   \prop_map_function:NN \g_\_\_hook_delayed_patches_prop { \use_i:nn }
58   \prop_gclear:N \g_\_\_hook_delayed_patches_prop
59   \cs_undefine:N \_\_hook_cmd_begindocument_code:
60 }
61 \g@addto@macro \g@kernel@after@begindocument
62 { \_\_hook_cmd_begindocument_code: }

(End of 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.

```

63 \cs_new_protected:Npn \_\_hook_cmd_try_patch:nn #1 #2
64 {
65   \_\_hook_debug:n
66   { \iow_term:x { ->~\string\begin{document}~try~cmd / #1 / #2. } }
67   \_\_hook_if_declared:nTF { cmd / #1 / #2 }
68   {
69     \_\_hook_debug:n
70     { \iow_term:n { .->~Giving~up:~hook~already~created. } }
71   }
72   {
73     \cs_if_exist:cT {#1}
74     { \exp_args:Nc \_\_hook_patch_command:Nnn {#1} {#1} {#2} }
75   }
76 }

(End of 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.

```

77 \cs_new_protected:Npn \_\_hook_patch_command:Nnn #1 #2 #3
78 {
79   \_\_hook_patch_debug:x { analyzing~'\token_to_str:N #1' }
80   \_\_hook_patch_debug:x { \token_to_str:N #1 = \token_to_meaning:N #1 }
81   \_\_hook_patch_check>NNnn \cs_if_exist:NTF #1 { undef }
82   {
83     \_\_hook_patch_debug:x { ++control~sequence~is~defined }
84     \_\_hook_patch_check>NNnn \token_if_macro:NTF #1 { macro }
85     {
86       \_\_hook_patch_debug:x { ++control~sequence~is~a~macro }

```

```

87         \_\_hook_patch_check:NNnn \_\_hook_if_public_command:NTF #1 { expl3 }
88         {
89             \_\_hook_patch_debug:x { +--+macro~is~not~private }
90             \robust@command@act
91                 \g_hook_patch_action_list_tl #1
92                 \_\_hook_retokenize_patch:Nnn { #1 {#2} {#3} }
93             }
94         }
95     }
96 }
```

And here's the auxiliary used above:

```

97 \cs_new_protected:Npn \_\_hook_patch_check:NNnn #1 #2 #3 #4
98 {
99     #1 #2 {#4}
100    {
101        \msg_error:nnxx { hooks } { cant-patch }
102        { \token_to_str:N #2 } {#3}
103    }
104 }
```

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.

```

105 \use:x
106 {
107     \prg_new_protected_conditional:Npnn
108         \exp_not:N \_\_hook_if_public_command:N ##1 { TF }
109     {
110         \exp_not:N \exp_last_unbraced:Nf
111         \exp_not:N \_\_hook_if_public_command:w
112         { \exp_not:N \cs_to_str:N ##1 }
113         \tl_to_str:n { _ _ } \s__hook_mark
114     }
115 }
116 \exp_last_unbraced:NNNNo
117 \cs_new_protected:Npn \_\_hook_if_public_command:w
118     #1 \tl_to_str:n { _ _ } #2 \s__hook_mark
119 {
120     \tl_if_empty:nTF {#2}
121     { \prg_return_true: }
122     { \prg_return_false: }
123 }
```

(End of 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.

```

124 \tl_gset:Nn \g_hook_patch_action_list_tl
125 {
126     { \c@if@DeclareRobustCommand \_\_hook_patch_DeclareRobustCommand:Nnn }
127     { \c@if@newcommand \_\_hook_patch_newcommand:Nnn }
```

```

128     { \__kernel_cmd_if_xparse:NTF \__hook_cmd_patch_xparse:Nnn }
129 }
```

(End of 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` with a `\c_false_bool` to indicate that there is no optional argument.

```

130 \cs_new_protected:Npn \__hook_patch_DeclareRobustCommand:Nnn #1
131 {
132     \exp_args:Nc \__hook_patch_DeclareRobustCommand_aux:Nnn
133     { \cs_to_str:N #1 ~ }
134 }
135 \cs_new_protected:Npn \__hook_patch_DeclareRobustCommand_aux:Nnn #1
136 {
137     \robust@command@chk@safe #1
138     { \@if@newcommand #1 }
139     { \use_i:nn }
140     { \__hook_patch_newcommand:Nnn }
141     { \__hook_patch_expand_redefine:NNnn \c_false_bool }
142     #1
143 }
```

(End of 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`.

```

144 \cs_new_protected:Npn \__hook_patch_newcommand:Nnn #1
145 {
146     \exp_args:NNc \__hook_patch_expand_redefine:NNnn \c_true_bool
147     { \c_underscore_str \cs_to_str:N #1 }
148 }
```

(End of definition for `__hook_patch_newcommand:Nnn`.)

`__hook_cmd_patch_xparse:Nnn` And for commands defined by the `xparse` commands use this for patching:

```

149 \cs_new_protected:Npn \__hook_cmd_patch_xparse:Nnn #1
150 {
151     \exp_args:NNc \__hook_patch_expand_redefine:NNnn \c_false_bool
152     { \cs_to_str:N #1 ~ code }
153 }
```

(End of 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 ≠ \csname#2\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 **tl** 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.

```
154 <|latexrelease>\IncludeInRelease{2023/06/01}{\_\_hook\_patch\_expand\_redefine:NNnn}  
155 <|latexrelease>                      {cmd~hooks~with~args}  
156 \cs_new_protected:Npn \_\_hook\_patch\_expand\_redefine:NNnn #1 #2 #3 #4  
157   {  
158     \_\_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>*).

```

159      \int_set:Nn \l__hook_patch_num_args_int
160      {
161          \exp_args:Nf \str_count:n { \cs_argument_spec:N #2 } / 2
162          \bool_if:NT #1 { -1 }
163      }

```

Now build two token lists:

`\l__hook_param_text_tl` 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_tl` will contain braced pairs of `\c__hook_hashes_tl<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_hashes_tl` here is to differentiate actual parameters in the macro from parameter tokens in the original definition of the macro. Later on, `\c__hook_hashes_tl` is either replaced by actual parameter tokens, or expanded into them.

```

164      \int_compare:nNnTF { \l__hook_patch_num_args_int } > { \c_zero_int }
165      {

```

We'll first check if the command has any parameter token in its definition (feeding it empty arguments), and set `__hook_exp_not:n` accordingly. `__hook_exp_not:n` will be used later to either leave `\c__hook_hashes_tl` or expand it, and also to remember the result of `__hook_if_has_hash:nTF` to avoid testing twice (the test can be rather slow).

```

166      \tl_set:Nx \l__hook_tmpa_tl { \bool_if:NTF #1 { [ ] } { { } } }
167      \int_step_inline:nnn { 2 } { \l__hook_patch_num_args_int }
168          { \tl_put_right:Nn \l__hook_tmpa_tl { { } } }
169      \exp_args:NNo \exp_args:No \__hook_if_has_hash:nTF
170          { \exp_after:wN #2 \l__hook_tmpa_tl }
171          { \cs_set_eq:NN \__hook_exp_not:n \exp_not:n }
172          { \cs_set_eq:NN \__hook_exp_not:n \use:n }
173      \cs_set_protected:Npn \__hook_tmp:w ##1 ##2
174          {
175              ##1 \l__hook_param_text_tl { \use:n ##2 }
176              ##1 \l__hook_replace_text_tl { \__hook_exp_not:n {##2} }
177          }

```

Here we'll conditionally add `[...]` around the first parameter:

```

178      \bool_if:NTF #1
179          { \__hook_tmp:w \tl_set:Nx { [ \c__hook_hashes_tl 1 ] } }
180          { \__hook_tmp:w \tl_set:Nx { { \c__hook_hashes_tl 1 } } }

```

Then, for every parameter from the second, just add it normally:

```

181      \int_step_inline:nnn { 2 } { \l__hook_patch_num_args_int }
182          { \__hook_tmp:w \tl_put_right:Nx { { \c__hook_hashes_tl ##1 } } }

```

Now, if the command has any parameter token in its definition (then `__hook_exp_not:n` is `\exp_not:n`), call `__hook_double_hashes:n` to double them, and replace every `\c__hook_hashes_tl` by `#`:

```

183     \tl_set:Nx \l__hook_replace_text_tl
184     { \exp_not:N #2 \exp_not:V \l__hook_replace_text_tl }
185     \tl_set:Nx \l__hook_replace_text_tl
186     {
187         \token_if_eq_meaning:NNTF \__hook_exp_not:n \exp_not:n
188         { \exp_args:NNV \exp_args:No \__hook_double_hashes:n }
189         { \exp_args:NV \exp_not:o }
190         \l__hook_replace_text_tl
191     }

```

And now, set a few auxiliaries for the case that the macro has parameters, so it won't be passed through `\unexpanded` (twice):

```

192     \cs_set_eq:NN \__hook_def_cmd:w \tex_gdef:D
193     \cs_set_eq:NN \__hook_exp_not:NN \prg_do_nothing:
194     }
195     {

```

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):

```

196     \tl_clear:N \l__hook_param_text_tl
197     \tl_set_eq:NN \l__hook_replace_text_tl #2
198     \cs_set_eq:NN \__hook_def_cmd:w \tex_xdef:D
199     \cs_set:Npn \__hook_exp_not:NN ##1 { \exp_not:N ##1 \exp_not:N }
200     }

```

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.

```

201     \group_begin:
202     \int_set:Nn \tex_escapechar:D { '\v' }
203     \use:x
204     {
205     \group_end:
206     \tl_set:Nx \exp_not:N \l__hook_patch_prefixes_tl
207     { \exp_not:N \__hook_make_prefixes:w \cs_prefix_spec:N #2 / / }
208     }

```

Here we redefine the hook to have the right number of arguments. Disabling the hook, undefining the `parameter` token list then calling `__hook_make_usable:nn` are enough to redefine the hook to the extent we want. Code stored in the hook and other metadata about it are not lost in the process.

```

209     \__hook_disable:n { cmd / #3 / #4 }
210     \cs_undefine:c { c__hook_cmd / #3 / #4_parameter_tl }
211     \__hook_make_usable:nn { cmd / #3 / #4 } { \l__hook_patch_num_args_int }

```

Now 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.

```

212     \use:e
213     {
214         \__hook_redefine_with_hooks:Nnnn \exp_not:N #2
215         \str_if_eq:nnTF {#4} { after }

```

```

216      { \use_ii_i:nn }
217      { \use:nn }
218      { {
219          \__hook_exp_not:NN \exp_not:N \UseHookWithArguments
220          { cmd / #3 / #4 } { \int_use:N \l__hook_patch_num_args_int }
221          \__hook_braced_parameter:n { cmd / #3 / #4 }
222      } }
223      { { } }
224      { \__hook_exp_not:NN \exp_not:V \l__hook_replace_text_tl }
225  }

```

Finally, update the hook code.

```

226      \__hook_update_hook_code:n { cmd / #3 / #4 }
227  }
228 <|latexrelease>\EndIncludeInRelease
229 <|latexrelease>\IncludeInRelease{2021/06/01}{\__hook_patch_expand_redefine:NNnn}
230 <|latexrelease>           {cmd-hooks-with-args}
231 <|latexrelease>\cs_gset_protected:Npn \__hook_patch_expand_redefine:NNnn #1 #2 #3 #4
232 <|latexrelease>  {
233 <|latexrelease>      \__hook_patch_debug:x { ++~command~can~be~patched~without~rescanning }
234 <|latexrelease>      \int_set:Nn \l__hook_patch_num_args_int
235 <|latexrelease>      {
236 <|latexrelease>          \exp_args:Nf \str_count:n { \cs_argument_spec:N #2 } / 2
237 <|latexrelease>          \bool_if:NT #1 { -1 }
238 <|latexrelease>      }
239 <|latexrelease>      \int_compare:nNnTF { \l__hook_patch_num_args_int } > { \c_zero_int }
240 <|latexrelease>      {
241 <|latexrelease>          \tl_set:Nx \l__hook_tmpa_tl { \bool_if:NTF #1 { [ ] } { { } } }
242 <|latexrelease>          \int_step_inline:nnn { 2 } { \l__hook_patch_num_args_int }
243 <|latexrelease>          { \tl_put_right:Nn \l__hook_tmpa_tl { { } } }
244 <|latexrelease>          \exp_args:NNo \exp_args:No \__hook_if_has_hash:nTF
245 <|latexrelease>          { \exp_after:WN #2 \l__hook_tmpa_tl }
246 <|latexrelease>          { \cs_set_eq:NN \__hook_exp_not:n \exp_not:n }
247 <|latexrelease>          { \cs_set_eq:NN \__hook_exp_not:n \use:n }
248 <|latexrelease>          \cs_set_protected:Npn \__hook_tmp:w ##1 ##2
249 <|latexrelease>          {
250 <|latexrelease>              ##1 \l__hook_param_text_tl { \use:n ##2 }
251 <|latexrelease>              ##1 \l__hook_replace_text_tl { \__hook_exp_not:n {##2} }
252 <|latexrelease>          }
253 <|latexrelease>          \bool_if:NTF #1
254 <|latexrelease>          { \__hook_tmp:w \tl_set:Nx { [ \c_hook_hash_t1 1 ] } }
255 <|latexrelease>          { \__hook_tmp:w \tl_set:Nx { { \c_hook_hash_t1 1 } } }
256 <|latexrelease>          \int_step_inline:nnn { 2 } { \l__hook_patch_num_args_int }
257 <|latexrelease>          { \__hook_tmp:w \tl_put_right:Nx { { \c_hook_hash_t1 ##1 } } }
258 <|latexrelease>          \tl_set:Nx \l__hook_replace_text_tl
259 <|latexrelease>          { \exp_not:N #2 \exp_not:V \l__hook_replace_text_tl }
260 <|latexrelease>          \tl_set:Nx \l__hook_replace_text_tl
261 <|latexrelease>          {
262 <|latexrelease>              \token_if_eq_meaning:NNTF \__hook_exp_not:n \exp_not:n
263 <|latexrelease>              { \exp_args:NNV \exp_args:No \__hook_double_hashes:n }
264 <|latexrelease>              { \exp_args:NV \exp_not:o }
265 <|latexrelease>              \l__hook_replace_text_tl
266 <|latexrelease>          }
267 <|latexrelease>          \cs_set_eq:NN \__hook_def_cmd:w \tex_gdef:D
268 <|latexrelease>          \cs_set_eq:NN \__hook_exp_not>NN \prg_do_nothing:

```

```

269 <{latexrelease}>      }
270 <{latexrelease}>      {
271   <{latexrelease}>      \tl_clear:N \l__hook_param_text_tl
272   <{latexrelease}>      \tl_set_eq:NN \l__hook_replace_text_tl #2
273   <{latexrelease}>      \cs_set_eq:NN \__hook_def_cmd:w \tex_xdef:D
274   <{latexrelease}>      \cs_set:Npn \__hook_exp_not:NN ##1 { \exp_not:N ##1 \exp_not:N }
275   <{latexrelease}>      }
276   <{latexrelease}>      \group_begin:
277     <{latexrelease}>      \int_set:Nn \tex_escapechar:D { '\/>
278     <{latexrelease}>      \use:x
279     <{latexrelease}>      {
280       <{latexrelease}>      \group_end:
281       <{latexrelease}>      \tl_set:Nx \exp_not:N \l__hook_patch_prefixes_tl
282       <{latexrelease}>      { \exp_not:N \__hook_make_prefixes:w \cs_prefix_spec:N #2 // }
283       <{latexrelease}>      }
284       <{latexrelease}>      \use:x
285       <{latexrelease}>      {
286         <{latexrelease}>      \__hook redefine_with_hooks:Nnnn \exp_not:N #2
287         <{latexrelease}>      \str_if_eq:nnTF {#4} { after }
288         <{latexrelease}>      { \use_i:i:nn }
289         <{latexrelease}>      { \use:nn }
290         <{latexrelease}>      { { \__hook_exp_not:NN \exp_not:N \UseHook { cmd / #3 / #4 } } }
291         <{latexrelease}>      { { } }
292         <{latexrelease}>      { \__hook_exp_not:NN \exp_not:V \l__hook_replace_text_tl }
293       <{latexrelease}>      }
294     <{latexrelease}>    }
295   <{latexrelease}> \EndIncludeInRelease

```

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.

```

296 \cs_new_protected:Npn \__hook redefine_with_hooks:Nnnn #1 #2 #3 #4
297   {
298     \l__hook_patch_prefixes_tl
299     \exp_after:wN \__hook_def_cmd:w
300     \exp_after:wN #1 \l__hook_param_text_tl
301     { #2 #4 #3 }
302   }

```

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.

```

303 \cs_new:Npn \__hook_make_prefixes:w / #1 /
304   {
305     \tl_if_empty:nF {#1}
306     {
307       \exp_not:c { tex_ \tl_trim_spaces:n {#1} :D }
308       \__hook_make_prefixes:w /
309     }
310   }

```

(End of 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
\_hook_if_has_hash:nTF found, it returns true, and false otherwise. The searching doesn't care about preserving
\_hook_if_has_hash:w groups or spaces: we can ignore those safely (braces are removed) so that searching is as
\_hook_if_has_hash_check:w fast as possible.
```

```
311 \prg_new_conditional:Npn \_hook_if_has_hash:n #1 { TF }
312   { \_hook_if_has_hash:w #1 ## \s_\_hook_mark }
313 \cs_new:Npn \_hook_if_has_hash:w #1
314   {
315     \tl_if_single_token:nTF {#1}
316     {
317       \token_if_eq_catcode:NNTF ## #1
318       { \_hook_if_has_hash_check:w }
319       { \_hook_if_has_hash:w }
320     }
321     { \_hook_if_has_hash:w #1 }
322   }
323 \cs_new:Npn \_hook_if_has_hash_check:w #1 \s_\_hook_mark
324   { \tl_if_empty:nTF {#1} { \prg_return_false: } { \prg_return_true: } }
```

(End of 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:n loops through the token list #1 and duplicates any catcode 6
\_hook_double_hashes:w token, and expands tokens \ifx-equal to \c_\_hook_hashes_t1, and leaves all other tokens
\_\_hook_double_hashes_output:N \notexpanded with \exp_not:N. Unfortunately pairs of explicit catcode 1 and
\_\_hook_double_hashes_stop:w catcode 2 character tokens are normalised to {1 and }1 because it's not feasible to ex-
\_\_hook_double_hashes_group:n pandably detect the character code (maybe it could be done using something along the
\_\_hook_double_hashes_space:w 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.

```
325 \cs_new:Npn \_hook_double_hashes:n #1
326   { \_hook_double_hashes:w #1 \q_\_hook_recursion_tail \q_\_hook_recursion_stop }
327 \cs_new:Npn \_hook_double_hashes:w #1 \q_\_hook_recursion_stop
328   {
329     \tl_if_head_is_N_type:nTF {#1}
330     { \_hook_double_hashes_output:N }
331     {
332       \tl_if_head_is_group:nTF {#1}
333       { \_hook_double_hashes_group:n }
334       { \_hook_double_hashes_space:w }
335     }
336     #1 \q_\_hook_recursion_stop
337   }
```

`_hook_double_hashes_output:N` checks for the end of the token list, then checks if the token is `\c__hook_hashes_t1`, and if so just leaves it.

```
338 \cs_new:Npn \_hook_double_hashes_output:N #1
339   {
340     \if_meaning:w \q_\_hook_recursion_tail #1
```

```

341      \__hook_double_hashes_stop:w
342      \fi:
343      \if:w ?
344          \if_meaning:w \c_hook_hash_tl #1 ! \fi:
345          \if_meaning:w \c_hook_hashes_tl #1 ! \fi:
346              ?
347      \else:

```

(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_hashes_tl.)

```

348      \use_i:nnnn
349      \fi:
350      \use:n
351      {

```

If #1 is not \c_hook_hashes_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.

```

352      \if_catcode:w ## \exp_not:N #1
353          \exp_after:wN \use_i:nnnn
354          \fi:
355          \use_none:n
356          { \exp_not:n { #1 #1 } }
357      }

```

If both previous tests returned `false`, then leave the token unexpanded and resume the loop.

```

358      \exp_not:N #1
359      \__hook_double_hashes:w
360      }
361 \cs_new:Npn \__hook_double_hashes_stop:w #1 \q__hook_recursion_stop { \fi: }
Dealing with spaces and grouped tokens is trivial:
362 \cs_new:Npn \__hook_double_hashes_group:n #1
363  { { \__hook_double_hashes:n {#1} } \__hook_double_hashes:w }
364 \exp_last_unbraced:NNo
365 \cs_new:Npn \__hook_double_hashes_space:w \c_space_tl
366  { ~ \__hook_double_hashes:w }

(End of 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`.

```

367 \cs_new_protected:Npn \__hook_retokenize_patch:Nnn #1 #2 #3
368 {
369     \str_if_eq:eeTF { \cs_argument_spec:N #1 } { }
370     { \__hook_patch_expand_redefine>NNnn \c_false_bool #1 {#2} {#3} }
371     {
372         \__hook_patch_debug:x { ..~command~can~only~be~patched~by~rescanning }

```

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 `\C`;
- 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`).

```

373     \tl_set:Nx \l__hook_tmpa_tl
374     {
375         \int_compare:nNnTF { \char_value_catcode:n {'\C} } = { 12 }
376             { \exp_not:N \makeatletter } { \exp_not:N \makeatother }
377         }
378     \tl_set:Nx \l__hook_tmpb_tl
379     {
380         \bool_if:NTF \l__kernel_expl_bool
381             { \ExplSyntaxOff } { \ExplSyntaxOn }
382         }
383     \use:x
384     {
385         \exp_not:N \__hook_try_patch_with_catcodes:Nnnnw
386             \exp_not:n { #1 {#2} {#3} }
387             { \prg_do_nothing: }
388             { \exp_not:V \l__hook_tmpa_tl } % @
389             { \exp_not:V \l__hook_tmpb_tl } % _:
390             {
391                 \exp_not:V \l__hook_tmpa_tl % @
392                 \exp_not:V \l__hook_tmpb_tl % _:
393             }
394     }
395     \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.

```

396     {
397         \msg_error:nnxx { hooks } { cant-patch }
398             { \c_backslash_str #2 } { retok }
399     }
400 }
401 }

```

(End of definition for `_hook_retokenize_patch:Nnn.`)

`_hook_try_patch_with_catcodes:Nnnnw`

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.

```
402 <latexrelease>\IncludeInRelease{2023/06/01}{\_hook_try_patch_with_catcodes:Nnnnw}
403 <latexrelease>                                {cmd~hooks~with~args}
404 \cs_new_protected:Npn \_hook_try_patch_with_catcodes:Nnnnw #1 #2 #3 #4
405 {
406     \quark_if_recursion_tail_stop_do:nn {#4} { \use:n }
407     \_hook_patch_debug:x { ++trying-to-patch-by-retokenization }
408     \_hook_cmd_if_scannable:NnTF {#1} {#4}
409     {
410         \_hook_patch_debug:x { +macro-can-be-retokenized-cleanly }
411         \_hook_patch_debug:x { ==retokenizing-macro-now }
412         \_hook_patch_retokenize:Nnnn #1 { cmd / #2 / #3 } {#3} {#4}
413         \use_i_delimit_by_q_recursion_stop:nw \use_none:n
414     }
415     {
416         \_hook_patch_debug:x { ---macro-cannot-be-retokenized-cleanly }
417         \_hook_try_patch_with_catcodes:Nnnnw #1 {#2} {#3}
418     }
419 }
420 <latexrelease>\EndIncludeInRelease
421 <latexrelease>\IncludeInRelease{2021/06/01}{\_hook_try_patch_with_catcodes:Nnnnw}
422 <latexrelease>                                {cmd~hooks~with~args}
423 \cs_gset_protected:Npn \_hook_try_patch_with_catcodes:Nnnnw #1 #2 #3 #4
424 <latexrelease> {
425     \quark_if_recursion_tail_stop_do:nn {#4} { \use:n }
426     \_hook_patch_debug:x { ++trying-to-patch-by-retokenization }
427     \_hook_cmd_if_scannable:NnTF {#1} {#4}
428     {
429         \_hook_patch_debug:x { +macro-can-be-retokenized-cleanly }
430         \_hook_patch_debug:x { ==retokenizing-macro-now }
431         \_hook_patch_retokenize:Nnnn #1 {#2} {#3} {#4}
432         \use_i_delimit_by_q_recursion_stop:nw \use_none:n
433     }
434     {
435         \_hook_patch_debug:x { ---macro-cannot-be-retokenized-cleanly }
436         \_hook_try_patch_with_catcodes:Nnnnw #1 {#2} {#3}
437     }
438 }
439 <latexrelease>\EndIncludeInRelease
```

(End of definition for `_hook_try_patch_with_catcodes:Nnnnw.`)

`\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`:

```
440 \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 of 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 `_0`, delimit the definition with `{_1}` and `{_2}`, and mark parameters with `#_6`. Everything else may be changed, but not these.

```
441 \cs_new_protected:Npn \_hook_patch_required_catcodes:
442   {
443     \char_set_catcode_escape:N \\%
444     \char_set_catcode_group_begin:N \{%
445     \char_set_catcode_group_end:N \}%
446     \char_set_catcode_parameter:N \#
447     % \int_set:Nn \tex_endlinechar:D { -1 }%
448     % \int_set:Nn \tex_newlinechar:D { -1 }%
449 }
```

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 *<parameter text>* and *<replacement text>*, then only `#` will have to stay in that list.

PhO: Actually now that we patch `\UseHook{cmd/foo/before}`, 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 of definition for `_hook_patch_required_catcodes`.)

`_hook_cmd_if_scannable:NnTF` 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 *<prefixes>*, *<parameter text>* and *<replacement text>* and arrange them as

$$\langle\textit{prefixes}\rangle\text{\def}\text{\kerneltmpDoNotUse}\langle\textit{parameter text}\rangle\{\langle\textit{replacement text}\rangle\}$$
3. rescan 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.

```
450 \prg_new_protected_conditional:Npnn \_hook_cmd_if_scannable:Nn #1 #2 { TF }
451   {
452     \cs_set_eq:NN \kerneltmpDoNotUse \scan_stop:
453     \cs_set_eq:NN \_hook_tmp:w \scan_stop:
454     \use:x
455     {
456       \cs_set:Npn \_hook_tmp:w
```

```

457         #####1 \tl_to_str:n { macro: } #####2 -> #####3 \s__hook_mark
458         { #####1 \def \kerneltmpDoNotUse #####2 {#####3} }
459         \tl_set:Nx \exp_not:N \l__hook_tmpa_tl
460         { \exp_not:N \__hook_tmp:w \token_to_meaning:N #1 \s__hook_mark }
461     }
462 \tl_rescan:nV { #2 \__hook_patch_required_catcodes: } \l__hook_tmpa_tl
463 \token_if_eq_meaning:NNTF #1 \kerneltmpDoNotUse
464   { \prg_return_true: }
465   { \prg_return_false: }
466 }

```

(End of definition for `__hook_cmd_if_scannable:NnTF`.)

`__hook_guess_arg_count:NN` Looks at the parameter text of a macro, and counts the parameters by looking at the number after a #, and checking if they are sequential. This macro assumes that all parameters are marked with hashes, and not other characters, and that there is no “trick parameter”.

```

467 \IfFileExists{2023/06/01}{\__hook_guess_arg_count:NN}
468 \IfFileExists{}{cmd~hooks~with~args}
469 \cs_new_protected:Npn \__hook_guess_arg_count:NN #1
470   {
471     \exp_after:wN \__hook_guess_arg_count:wN
472     \token_to_meaning:N #1 \s__hook_mark
473   }
474 \exp_last_unbraced:NNNNo
475 \cs_new_protected:Npx \__hook_guess_arg_count:wN
476   #1 { \tl_to_str:n { macro: } } #2 \s__hook_mark #3
477   {
478     \int_set:Nn #3
479     {
480       \exp_not:N \__hook_guess_arg_count:nw { 0 } #2
481       \c_hash_str 0 \s__hook_mark
482     }
483   }
484 \use:e
485   { \cs_new:Npn \exp_not:N \__hook_guess_arg_count:nw #1 #2 \c_hash_str #3 }
486   {
487     \int_compare:nNnTF { #1 + 1 } = {#3}
488     { \__hook_guess_arg_count:nw {#3} }
489     { #1 \__hook_use_none_delimit_by_s_mark:w }
490   }
491 \EndIncludeInRelease
492 \IfFileExists{2021/06/01}{\__hook_guess_arg_count:NN}
493 \IfFileExists{}{cmd~hooks~with~args}
494 \cs_undefine:N \__hook_guess_arg_count:NN
495 \EndIncludeInRelease

```

(End of definition for `__hook_guess_arg_count:NN`, `__hook_guess_arg_count:wN`, and `__hook_guess_arg_count:nw`.)

`__hook_patch_retokenize:Nnnn` Then, if `__hook_cmd_if_scannable:NnTF` returned true, we can go on and patch the command.

```

496 \IfFileExists{2023/06/01}{\__hook_patch_retokenize:Nnnn}
497 \IfFileExists{}{cmd~hooks~with~args}

```

```

498 \cs_new_protected:Npn \__hook_patch_retokenize:Nnnn #1 #2 #3 #4
499 {
500 %   Here, when patching by retokenization, we can only guess the number
501 %   of arguments of the macro.
502 % \changes{v1.0h}{2023/05/21}
503 %           {Changes to allow support arguments in cmd hooks (cmd-args).}
504 % \begin{macrocode}
505 \__hook_guess_arg_count:NN #1 \l__hook_patch_num_args_int

```

Then we redefine the hook to have the right number of arguments. Disabling the hook, undefining the `parameter` token list then calling `__hook_make_usable:nn` are enough to redefine the hook to the extent we want. Code stored in the hook and other metadata about it are not lost in the process.

```

506 \__hook_disable:n {#2}
507 \cs_undefine:c { c__hook_##2_parameter_tl }
508 \__hook_make_usable:nn {#2} { \l__hook_patch_num_args_int }
509 \tl_set:Ne \l__hook_tmpa_tl
510 { \exp_args:Ne \tl_to_str:n { \__hook_braced_parameter:n {#2} } }
511 \use:x
512 {
513     \str_replace_all:Nnn \exp_not:N \l__hook_tmpa_tl
514     { ##### } { \c_hash_str }
515 }

```

Then, make make some things `\relax` to avoid lots of `\noexpand` below.

```

516 \cs_set_eq:NN \kerneltmpDoNotUse \scan_stop:
517 \cs_set_eq:NN \__hook_tmp:w \scan_stop:
518 \use:x
519 {

```

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.

```

520 \cs_set:Npn \__hook_tmp:w
521     #####1 \tl_to_str:n { macro: } #####2 -> #####3 \s__hook_mark
522 {
523     #####1 \def \kerneltmpDoNotUse #####
524     {
525         \str_if_eq:nnT {#3} { before }
526         {
527             \token_to_str:N \UseHookWithArguments {#2}
528             { \int_use:N \l__hook_patch_num_args_int }
529             \l__hook_tmpa_tl
530         }
531     #####3
532     \str_if_eq:nnT {#3} { after }
533     {
534         \token_to_str:N \UseHookWithArguments {#2}

```

```

535           { \int_use:N \l__hook_patch_num_args_int }
536           \l__hook_tmpa_tl
537       }
538   }
539 }
```

Now we just have to get the `\meaning` of the command being patched and pass it through the meat grinder above.

```

540     \tl_set:Nx \exp_not:N \l__hook_tmpa_tl
541     { \exp_not:N \__hook_tmp:w \token_to_meaning:N #1 \s__hook_mark }
542 }
```

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.

```
543     \tl_rescan:nV { #4 \__hook_patch_required_catcodes: } \l__hook_tmpa_tl
```

And to close, copy the newly-defined command into the old name and the patching is finally completed:

```
544     \cs_gset_eq:NN #1 \kerneltmpDoNotUse
```

Finally, update the hook code.

```

545     \__hook_update_hook_code:n {#2}
546   }
547   ⟨latexrelease⟩\EndIncludeInRelease
548   ⟨latexrelease⟩\IncludeInRelease{2021/06/01}{\__hook_patch_retokenize:Nnnn}
549   ⟨latexrelease⟩          {cmd~hooks~with~args}
550   ⟨latexrelease⟩\cs_gset_protected:Npn \__hook_patch_retokenize:Nnnn #1 #2 #3 #4
551   ⟨latexrelease⟩  {
552   ⟨latexrelease⟩    \cs_set_eq:NN \kerneltmpDoNotUse \scan_stop:
553   ⟨latexrelease⟩    \cs_set_eq:NN \__hook_tmp:w \scan_stop:
554   ⟨latexrelease⟩    \use:x
555   ⟨latexrelease⟩    {
556   ⟨latexrelease⟩      \cs_set:Npn \__hook_tmp:w
557   ⟨latexrelease⟩        #####1 \tl_to_str:n { macro: } #####2 -> #####3 \s__hook_mark
558   ⟨latexrelease⟩    {
559   ⟨latexrelease⟩      #####1 \def \kerneltmpDoNotUse #####2
560   ⟨latexrelease⟩        {
561   ⟨latexrelease⟩          \str_if_eq:nnT {#3} { before }
562   ⟨latexrelease⟩            { \token_to_str:N \UseHook { cmd / #2 / #3 } }
563   ⟨latexrelease⟩          #####3
564   ⟨latexrelease⟩          \str_if_eq:nnT {#3} { after }
565   ⟨latexrelease⟩            { \token_to_str:N \UseHook { cmd / #2 / #3 } }
566   ⟨latexrelease⟩        }
567   ⟨latexrelease⟩    }
568   ⟨latexrelease⟩    \tl_set:Nx \exp_not:N \l__hook_tmpa_tl
569   ⟨latexrelease⟩    { \exp_not:N \__hook_tmp:w \token_to_meaning:N #1 \s__hook_mark }
570   ⟨latexrelease⟩  }
571   ⟨latexrelease⟩  \tl_rescan:nV { #4 \__hook_patch_required_catcodes: } \l__hook_tmpa_tl
572   ⟨latexrelease⟩  \cs_gset_eq:NN #1 \kerneltmpDoNotUse
573   ⟨latexrelease⟩ }
574   ⟨latexrelease⟩\EndIncludeInRelease
```

(End of definition for `__hook_patch_retokenize:Nnnn`.)

4.6 Messages

```

575 <latexrelease>\IncludeInRelease{2023/06/01}{wrong-cmd-hook}%
576 <latexrelease>                                {Standardise-generic-hook-names}
577 <latexrelease>\EndIncludeInRelease
578 <latexrelease>\IncludeInRelease{2021/06/01}{wrong-cmd-hook}%
579 <latexrelease>                                {Standardise-generic-hook-names}
580 <latexrelease>\msg_new:nnn { hooks } { wrong-cmd-hook }
581 <latexrelease>  {
582 <latexrelease>    Generic~hook~`cmd/#1/#2'~is~invalid.
583 <latexrelease>%     The~hook~should~be~`cmd/#1/before'~or~`cmd/#1/after'.
584 <latexrelease>  }
585 <latexrelease>  {
586 <latexrelease>    You~tried~to~add~a~generic~hook~to~command~\iow_char:N \\#1,~but~`#2'~
587 <latexrelease>    is~an~invalid~component.~Only~`before'~or~`after'~are~allowed.
588 <latexrelease>  }
589 <latexrelease>\EndIncludeInRelease
590 \msg_new:nnn { hooks } { cant-patch }
591  {
592   Generic~hooks~cannot~be~added~to~`#1'.
593  }
594  {
595   You~tried~to~add~a~hook~to~`#1',~but~LaTeX~was~unable~to~
596   patch~the~command~because~it~\_\_hook\_unpatchable\_cases:n {#2}.
597  }
598 \cs_new:Npn \_\_hook_unpatchable_cases:n #1
599  {
600   \str_case:nn {#1}
601   {
602     { undef } { doesn't-exist }
603     { macro } { is-not-a-macro }
604     { expl3 } { is-a-private-expl3-macro }
605     { retok } { can't-be-retokenized-cleanly }
606   }
607  }
608 <latexrelease>\IncludeInRelease{0000/00/00}{ltcmandhooks}%
609 <latexrelease>                                {The~hook~management~system~for~commands}
610 <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.

```

611 <latexrelease>\cs_set_eq:NN \_\_hook_cmd_begindocument_code: \prg_do_nothing:
612 <latexrelease>
613 <latexrelease>\EndModuleRelease
614 \ExplSyntaxOff
615 </2ekernel | latexrelease>
616 <@@=〉

```

File j

ltalloc.dtx

1 Counters

This section deals with counter and other variable allocation.

1 {*2ekernel}

The following are from plain T_EX:

\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 of 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 of definition for \c@Mi and others.)

\c@tempcnta Scratch count registers used by L^AT_EX kernel commands.

\c@tempcntb 7 \newcount\c@tempcnta

8 \newcount\c@tempcntb

(End of definition for \c@tempcnta and \c@tempcntb.)

\c@if@tempswa General boolean switch used by L^AT_EX kernel commands.

9 \newif\c@if@tempswa

(End of definition for \c@if@tempswa.)

\c@tempdima Scratch dimen registers used by L^AT_EX kernel commands.

\c@tempdimb 10 \newdimen\c@tempdima

\c@tempdimc 11 \newdimen\c@tempdimb

12 \newdimen\c@tempdimc

(End of definition for \c@tempdima, \c@tempdimb, and \c@tempdimc.)

\@tempboxa Scratch box register used by L^AT_EX kernel commands.
 ¹³ \newbox\@tempboxa
 (End of definition for \@tempboxa.)

\@tempskipa Scratch skip registers used by L^AT_EX kernel commands.
 \@tempskipb
 ¹⁴ \newskip\@tempskipa
 ¹⁵ \newskip\@tempskipb
 (End of definition for \@tempskipa and \@tempskipb.)

\@temptokena Scratch token register used by L^AT_EX kernel commands.
 ¹⁶ \newtoks\@temptokena
 (End of definition for \@temptokena.)

\@flushglue Glue used for \right- & \leftskip = 0pt plus 1fil
 ¹⁷ \newskip\@flushglue \@flushglue = 0pt plus 1fil
 (End of definition for \@flushglue.)
 ¹⁸ ⟨/2ekernel⟩

File k

ltcntrl.dtx

1 Program control structure

This section defines a number of control structure macros, such as while-loops and for-loops.

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

```
1  {*2ekernel}
2  \message{control,}

\@whilenum TEST \do {BODY}
\@whiledim TEST \do {BODY} : These implement the loop
    while TEST do BODY od
    where TEST is a TeX \ifnum or \ifdim test, respectively.
    They are optimized for the normal case of TEST initially false.

\@whilesw SWITCH \fi {BODY} : Implements the loop
    while SWITCH do BODY od
    Optimized for normal case of SWITCH initially false.

\@for NAME := LIST \do {BODY} : Assumes that LIST expands to A1,A2,
... ,An .
    Executes BODY n times, with NAME = Ai on the i-th iteration.
    Optimized for the normal case of n = 1. Works for n=0.

\@tfour NAME := LIST \do {BODY}
    if, before expansion, LIST = T1 ... Tn where each Ti is a
    token or {...}, then executes BODY n times, with NAME = Ti
    on the i-th iteration. Works for n=0.
```

NOTES: 1. These macros use no `\@temp` sequences.
2. These macros do not work if the body contains anything that looks syntactically to TeX like an improperly balanced `\if` `\else` `\fi`.

```
\@whilenum TEST \do {BODY} ==
BEGIN
  if TEST
  then BODY
    \@iwhilenum{TEST \relax BODY}
END

\@iwhilenum {TEST BODY} ==
BEGIN
  if TEST
  then BODY
```

```

        \cnextwhile = def(\@iwhilenum)
else  \cnextwhile = def(\@whilenoop)
fi
\cnextwhile {TEST BODY}
END

\@whilesw SWITCH \fi {BODY} ==
BEGIN
if SWITCH
then BODY
\@iwhilesw {SWITCH BODY}\fi
fi
END

\@iwhilesw {SWITCH BODY} \fi ==
BEGIN
if SWITCH
then BODY
\cnextwhile = def(\@iwhilesw)
else \cnextwhile = def(\@whileswnoop)
fi
\cnextwhile {SWITCH BODY} \fi
END

```

End of historical L^AT_EX 2.09 comments.

```

\@whilenoop
\@whilenum
\@iwhilenum
3 \long\def\@whilenum#1\do #2{\ifnum #1\relax #2\relax\@iwhilenum{#1\relax
4      #2\relax}\fi}
5 \long\def\@iwhilenum#1{\ifnum #1\expandafter\@iwhilenum
6      \else\expandafter\@gobble\fi{#1}}

```

(End of 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 of definition for \@whiledim and \@iwhiledim.)

```

\@whileswnoop
\@whilesw
\@iwhilesw
10 \long\def\@whilesw#1\fi{#1#2\@iwhilesw{#1#2}\fi\fi}
11 \long\def\@iwhilesw#1\fi{#1\expandafter\@iwhilesw
12      \else\@gobbletwo\fi{#1}\fi}

```

(End of definition for \@whileswnoop, \@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 of definition for \@nnil.)

\@empty

¹⁴ \def\@empty{}

(End of definition for \@empty.)

```

\@fornoop
15 \long\def\@fornoop#1\@@#2#3{%
(End of 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 of 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 of 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 of 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 of 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 of 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 of 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 of 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 of 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 of 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{^J}%
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 of definition for \GenericError.)

\PackageError \PackageWarning \PackageWarningNoLine \PackageInfo \ClassError \ClassWarning \ClassWarningNoLine \ClassInfo	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 of 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 of definition for `\ClassNote` and others.)

<code>\@latex@error</code>	Errors and other info, for use in the L ^A T _E X core.
<code>\@latex@warning</code>	
<code>\@latex@warning@no@line</code>	
<code>\@latex@info</code>	
<code>\@latex@info@no@line</code>	

```

163 \gdef\@latex@error#1#2{%
164     \GenericError{%
165         \space\space\space\@spaces\@spaces\@spaces
166     }{%
167         LaTeX Error: #1%
168     }{%

```

```

169      See the LaTeX manual or LaTeX Companion for explanation.%  

170  }{#2}%
171 }
172 \def\@latex@warning#1{%
173   \GenericWarning{%
174     \space\space\space\@spaces\@spaces\@spaces
175   }{%
176     LaTeX Warning: #1%
177   }%
178 }
179 \def\@latex@warning@no@line#1{%
180   \@latex@warning{#1\@gobble}}
181 \def\@latex@info#1{%
182   \GenericInfo{%
183     \@spaces\@spaces\@spaces
184   }{%
185     LaTeX Info: #1%
186   }%
187 }
188 \def\@latex@info@no@line#1{%
189   \@latex@info{#1\@gobble}}

```

\@font@warning and \@font@info are defined later since they have to be redefined by the `tracefn` package.

```

def\@font@warning#1{%
  \GenericWarning{%
    {(font)}\@spaces\@spaces}%
    {Font Warning: #1}%
}
def\@font@info#1{%
  \GenericInfo{%
    (font)\space\@spaces
  }{%
    Font Info: #1%
  }%
}

```

(End of definition for \@latex@error and others.)

\@latex@note These are “info” messages that display on the terminal not just in the transcript.

```

\@latex@note@no@line 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}}
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 of 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 of definition for `\c@errorcontextlines`.)

`\on@line` The message ‘on input line *n*’.

```

214 \def\on@line{ on input line \the\inputlineno}

```

(End of 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 of definition for `\@warning`, `\@@warning`, and `\@latexerr`.)

`\@spaces` Four spaces.

```

218 \def\@spaces{\space\space\space\space}

```

(End of 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 of 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 of 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 of definition for \@nolnerr.)

- \@nocounterr Generated by \setcounter, \addtocounter or \newcounter if applied to an undefined counter  $\langle cnt \rangle$ .

\@nocnterr Obsolete error message generated in L<sup>A</sup>T<sub>E</sub>X2.09 by \setcounter, \addtocounter or \newcounter for undefined counter. DO NOT use for L<sup>A</sup>T<sub>E</sub>X2 <sub>$\varepsilon$</sub>  it MIGHT vanish! Use \@nocounterr{\mathit{cnt}} instead.

```

240 \gdef\@nocnterr#1{%
241 \@latex@error{No counter '#1' defined}\@eha}
242 \gdef\@nocnterr{\@nocounterr?}

```

(End of 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 of 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 of 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 of definition for \@badend.)

\@badmath Called by \[, \], \{ or \} when used in wrong mode.

```
251 \gdef\@badmath{%
252 \@latex@error{Bad math environment delimiter}\@eha}
```

(End of 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 of 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 of definition for \@badpoptabs.)

\@badtab Called by \>, \+ , \- or \< when stepping to an undefined tab.

```
258 \gdef\@badtab{%
259 \@latex@error{Undefined tab position}\@ehd}
```

(End of 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 of 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 of 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 of 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 of 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 of definition for \@latexbug.)

\@badcrerr This error was removed and replaced by \@nolnerr.

```
277 \%def\@badcrerr {\@latex@error{Bad use of \protect\\}\@ehc}
```

(End of 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 of 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 of 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 of 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 of definition for \@invalidchar.)

As well as the above error commands some error messages are directly coded to save space. The messages already present in L<sup>A</sup>T<sub>E</sub>X2.09 include:

Environment --- undefined

Issued by \begin for undefined environment.

Tab overflow

Occurs in \= when maximum number of tabs exceeded.

\< in mid line

Occurs in \< when it appears in middle of line.

Float(s) lost

In output routine, caused by a float environment or \marginpar occurring in inner vertical mode.

### 1.3 Tracing

The **trace** package implements the commands \traceon and \traceoff that work similar to \tracingall but skip certain code blocks that produce a lot of tracing output being of no interest during debugging (for example loading a font). Code blocks that should be hidden during tracing need to be surrounded by the macros \conditionally@traceoff and \conditionally@traceon.

For the kernel code the **trace** package then redefines a number of macros to include this tracing support.

However, in order to allow any macro package to react to \traceon we also provide dummy definitions for the two commands in the kernel so that they can be used by external packages without the need to distinguish between **trace** being loaded or not.

\conditionally@traceon These are only dummy definitions. For details see the **trace** package.

\conditionally@traceoff

289 \let\conditionally@traceon\@empty

290 \let\conditionally@traceoff\@empty

(End of definition for \conditionally@traceon and \conditionally@traceoff.)

291

# 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 of definition for `\@setpar` and `\@par`.)

`\@restorepar` Restore from a short-term change to `\par`.  
`\@restorepar`    5 `\def\@restorepar{\def\par{\@par}}`  
`\@restorepar`    6 `{/2ekernel}`

(End of 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.

## 1 Introduction

The building of paragraphs in the T<sub>E</sub>X 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 T<sub>E</sub>Xbook [?, chap. 14] (for the full truth you may even have to study the program code).

### 1.1 The default processing done by the engine

T<sub>E</sub>X 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, T<sub>E</sub>X will immediately back up (i.e., the character or command is read later again), adds a `\parskip` glue to the current vertical list unless the list is empty, switches to horizontal mode, starts its special “start of paragraph processing” and only then rereads the character or command that caused the mode change.<sup>19</sup>

This “start of paragraph processing” first adds an empty box at the start of the horizontal list of width `\parindent` (which represents the paragraph indentation) unless the paragraph was started with `\noindent` in which case no such box is added<sup>20</sup>. It then reads and processes all tokens stored in the special engine token register `\everypar`. After that it reads and processes whatever has caused the paragraph to start.

Thus out of the box, T<sub>E</sub>X 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 La<sub>T</sub>e<sub>X</sub> 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 T<sub>E</sub>X), 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.

The process when the paragraph ends has different mechanisms and interfaces. A paragraph ends when the engine primitive `\par` is called while T<sub>E</sub>X 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 T<sub>E</sub>X is in, e.g., the `\par` in `\hbox{a\par b}` is ignored, but `$a\par b$` would complain.

<sup>19</sup>Already not quite true: the command `\noindent` starts the paragraph but influences the special processing by suppressing the paragraph indentation box normally inserted by it.

<sup>20</sup>That’s a bit different from placing a zero-sized box!

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  $\LaTeX$  kernel; if some package uses low-level  $\vbox$ es without adding this precaution the  $\TeX$  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  $\LaTeX$  or  $$$...$$$  in plain  $\TeX$ , then  $\TeX$  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 TeX 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<sup>A</sup>T<sub>E</sub>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 TeX, e.g., we install special code inside `\everypar` to provide hooks at the beginning and we redefine `\par` to do some special processing when appropriate to install hooks at the end of the paragraph.

In order to make this work, we have to ensure that package use of `\everypar` is not overwriting our code. This is done through a trick: we basically hide the real `\everypar` from the packages and offer them a new token register (with the same name). So if they install their own code it doesn’t overwrite ours. Our code then inserts the new `\everypar` at the right place inside the process so that it looks as if it was the primitive `\everypar`.<sup>21</sup>

At the end of the paragraph it would be great if we could use a similar trick. However, due to the fact that TeX inserts the token `\par` (that doesn’t have a defined meaning) we can’t hide “the real thing™” and offer the package an indistinguishable alternate.

Fortunately, L<sup>A</sup>T<sub>E</sub>X has already redefined `\par` for its own purposes. As a result there aren’t many packages that attempt to change `\par`, because without a lot of extra care that would fail miserably. But the bottom line is that, if you load a package that alters `\par` then the end of paragraph hooks are most likely not executing while that redefinition is active.<sup>22</sup>

---

<sup>21</sup>Ideally, `\everypar` wouldn’t be used at all by packages and instead they would simply write their code into the hooks now offered by the kernel. However, while this is the longterm goal and clearly an improvement (because then the packages do no longer need to worry about getting their code overwritten or needing to account for already existing code in `\everypar`), this will not happen overnight. For that reason support for this legacy method is retained.

<sup>22</sup>Similarly to the `\everypar` situation, the remedy is that such packages stop doing this and instead add their alterations into the paragraph hooks now provided.

## 2.1 The provided hooks

---

**para/before** The following four public hooks are defined and executed for each paragraph:  
**para/begin**  
**para/end**  
**para/after** **para/before** This hook is executed after the kernel hook `\@kernel@before@para@before` (discussed below) in vertical mode immediately after TeX has contributed `\parskip` to the vertical list and before the actual paragraph processing in horizontal mode starts.

This hook should either not produce any typeset material or add only vertical material. If it starts a paragraph an error is generated. The reason is that we are in the starting process of processing a paragraph and so this would lead to endless recursion.<sup>23</sup>

**para/begin** This hook is executed after the kernel hook `\@kernel@before@para@begin` (discussed below) in horizontal mode immediately before the indentation box is placed (if there is any, i.e., if the paragraph hasn't been started with `\noindent`).

The indentation box to be typeset is available to the hook as `\IndentBox` and its automatic placement (after the hook is executed) can be prevented through `\OmitIndent`. More precisely `\OmitIndent` voids the box.

The indentation box is then typeset directly after the hook execution by something equivalent to `\box\IndentBox` followed by the current content of the token register `\everypar` that it is available to the kernel or to packages (that run some legacy code).

One has to be careful not to add any code to the hook that starts its own paragraph (e.g., by adding a `\parbox` or a `\marginpar` inside) because that would call the hook inside again (as a new paragraph is started there) and thus lead to an endless recursion ending only after exhausting the available memory. This can only be done by making sure that is not executed for the inner paragraphs (or at least not recursively forever).

**para/end** This hook is executed at the end of a paragraph when TeX is ready to return to vertical mode and after it has removed the last horizontal glue (but not any kerns) placed on the horizontal list. The code is still executed in horizontal mode so it is possible to add further horizontal material at this point, but it should not alter the mode (even a temporary exit from horizontal mode would create chaos—any attempt will cause an error message)! After the hook has ended the kernel hook `\@kernel@after@para@end` is executed and then TeX returns to vertical mode.

The hook is offered as public hook, but because of the requirement to stay within horizontal mode one needs to be careful in what is placed into the hook.<sup>24</sup>

This hook is implemented as a reversed hook.

**para/after** This hook is executed directly after TeX has returned to vertical mode and after any material that migrated out of the horizontal list (e.g., from a `\vadjust`) has processed.

---

<sup>24</sup>Maybe we should guard against that, but it would be rather tricky to implement as mode changes can happen across group boundaries so one would need to keep a private stack just for that. Well, something to ponder.

This hook should either not produce any typeset material or add only vertical material. However, for this hook starting a new paragraph is not a disaster so that it isn't prevented.

This hook is implemented as a reversed hook.

Once that hook code has been processed the kernel hook `\@kernel@after@para@after` is executed as the final action of the paragraph processing.

---

```
\@kernel@before@para@before
\@kernel@after@para@after
\@kernel@before@para@begin
\@kernel@after@para@end
```

---

As already mentioned above there are also four kernel hooks that are executed at the start and end of the processing.

`\@kernel@before@para@before` For future extensions, not currently used by the kernel.

`\@kernel@after@para@after` For future extensions, not currently used by the kernel.

`\@kernel@before@para@begin` Used by the kernel to implement tagging. This hook is executed at the very beginning of a paragraph after TeX has switched to horizontal mode but before any indentation box got added or any `\everypar` was run.

It should not generate typeset material that could alter the position. Note that it should never leave hmode, otherwise you will end with a loop! We could guard against this, but since it is an internal kernel hook that shouldn't be touched this isn't checked.

`\@kernel@after@para@end` Used by the kernel to implement tagging. It is executed directly after the public `para/end` hook. After it there is a quick check that we are still in horizontal mode, i.e., that the public hook has not mistakenly ended horizontal mode prematurely (this is an incomplete check just testing the mode and could perhaps be improved (at the cost of speed)).

## 2.2 Altered and newly provided commands

---

`\par`  
`\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<sup>A</sup>T<sub>E</sub>X `\endgraf` (with that behavior) was originally also available.

With the new paragraph handling in L<sup>A</sup>T<sub>E</sub>X, ending a paragraph means a bit more than just calling the engine's paragraph builder: the process also has to add any hook code for the end of a paragraph. Thus `\endgraf` was changed to provide this additional functionality (along with `\par` remaining subject to its current meaning).

The expl3 name for this functionality is `\para_end:`.

**Note:** *The next two commands are still under discussion and may slightly change their semantics (as described in the document) and/or their names between now and the 2021 Spring release!*

---

`\OmitIndent`  
`\para omit indent:`

Inside the `para/begin` hook one can use this command to suppress the indentation box at the start of the paragraph. (Technically it is possible to use this command outside the hook as well, but this should not be relied upon.) The box itself remains available for use.

The `expl3` name for the function is `\para omit indent:`.

---

`\IndentBox`  
`\g para indent box`

The box register holding the indentation box for the paragraph is available for inspection (or changes) inside hooks. It remains available even if the `\OmitIndent` command was used; in that case it will just not be automatically placed.

The `expl3` name for the box register is `\g para indent box`.

---

`\RawIndent`  
`\para raw indent:`  
`\RawNoindent`  
`\para raw noindent:`  
`\RawParEnd`  
`\para raw end:`

`\RawIndent hmode material \RawParEnd`  
`\RawNoindent hmode material \RawParEnd`

The commands `\RawIndent` and `\RawNoindent` are not meant for normal paragraph building (where the result is a textual paragraph in the traditional meaning of the word), but for special cases where TeX's low-level algorithm is used to achieve special effects, but where the result is not a "paragraph".

They are called "raw", because they bypass L<sup>A</sup>T<sub>E</sub>X's hook mechanism for paragraphs and simply invoke the low-level TeX algorithm. I.e., they are like the original TeX primitives `\indent` and `\noindent` (that is they execute no hooks other than `\everypar`) except that they can only be used in vertical mode and generate an error if found elsewhere.

To avoid issues a paragraph started by them should always be ended by `\RawParEnd`<sup>25</sup> and not by `\par` (or a blank line), because the latter will execute hooks which then have no counterpart at the beginning of the paragraph. It is the responsibility of the programmer to make sure that they are properly paired. This also means that one should not put arbitrary user content between these commands if that content could contain stray `\pars`.

The `expl3` names for the functions are `\para raw indent:`, `\para raw indent:` and `\para raw end:`.

## 2.3 Examples

None of the examples in this section are meant for real use as they are far too simple-minded but they should give some ideas of what could be possible if a bit more care is applied.

### 2.3.1 Testing the mechanism

The idea is to output for each paragraph encountered some information: a paragraph sequence number, a level number in roman numerals, the environment in which this paragraph appears, and the line number where the start or end of the paragraph is, e.g., something like

```
PARA: 1-i start (document env. on input line 38)
PARA: 1-i end (document env. on input line 38)
PARA: 2-ii start (minipage env. on input line 40)
PARA: 3-ii start (minipage env. on input line 40)
```

```

PARA: 3-ii end (minipage env. on input line 40)
PARA: 2-i end (document env. on input line 41)

```

As you can see paragraph 2 starts on line 40 and ends on 41 and inside a minipage started paragraph 3 (start and end on line 40). If you run this on some document you will find that L<sup>A</sup>T<sub>E</sub>X considers more things “a paragraph” than you have probably thought.

This was generated by the following hook code:

```

\newcounter{paracnt} % sequence counter
\newcounter{paralevel} % level counter

```

To support paragraph nesting we need to maintain a stack of the sequence numbers. This is most easily done using `expl3` functions, so we switch over. This is not a very general implementation, just enough for what we need and a bit of L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub>  thrown in as well. When popping, the result gets stored in `\paracntvalue` and the `\ERROR` should never happen because it means we have tried to pop from an empty stack.

```

\ExplSyntaxOn
\seq_new:N \g_para_seq
\cs_new:Npn \ParaPush
 {\seq_gpush:No \g_para_seq {\the\value{paracnt}}}
\cs_new:Npn \ParaPop {\seq_gpop:NNF \g_para_seq \paracntvalue \ERROR }
\ExplSyntaxOff

```

At the start of the paragraph increment both sequence counter and level and also save the then current sequence number on our stack.

```

\AddToHook{para/begin}{%
 \stepcounter{paracnt}\stepcounter{paralevel}%
 \ParaPush
}

```

To display the sequence number we `\typeout` the current sequence and level number. The command `\@currenvir` gives us the current environment and `\on@line` produces a space and the current input line number.

```

\typeout{PARA: \arabic{paracnt}-\roman{paralevel} start
(\@currenvir\space env.\on@line)}%

```

We also typeset the sequence number as a tiny red number in a box that takes up no horizontal space. This helps us seeing where L<sup>A</sup>T<sub>E</sub>X sees the start and end of the paragraphs in the document.

```

\llap{\color{red}\tiny\arabic{paracnt}\ }%
}

```

At the end of the paragraph we display sequence number and level again. The level counter has the correct value but we need to retrieve the right sequence value by popping it off the stack after which it is available in `\paracntvalue` the way we have set this up above.

```

\AddToHook{para/end}{%
 \ParaPop
 \typeout{PARA: \paracntvalue-\roman{paralevel} end \space\space
(\@currenvir\space env.\on@line)}%
}

```

We also typeset again a tiny red number with that value, this time sticking out to the right.<sup>26</sup> We also decrement the level counter since our level has finished.

```
\rlap{\color{red}\tiny\ \paracntvalue}%
\addtocounter{paralevel}{-1}%
}
\makeatother
```

### 2.3.2 Mark the first paragraph of each `itemize`

The code for this is rather simple. We supply some code that is executed only once inside a hook at the start of each `itemize`. We explicitly change the color back and forth so that we don't introduce grouping around the paragraph.

```
\AddToHook{env/itemize/begin}{%
\AddToHookNext{para/begin}{\color{blue}}%
\AddToHookNext{para/end}{\color{black}}%
}
```

As a result the first paragraph of each `itemize` will appear in blue.

## 2.4 Some technical notes

The code tries hard to be transparent for package code, but of course any change means that there is a potential for breaking other code. So in section we collect a few cases that may be of importance if low-level code is dealing with paragraphs that are now behaving slightly differently. The notes are from issues we observed and will probably grow over time.

### 2.4.1 Glue items between paragraphs (found with `fancypar`)

In the past L<sup>A</sup>T<sub>E</sub>X placed two glue items between two consecutive paragraphs, e.g.,

```
text1 \par text2 \par
```

would show something like

```
\glue(\parskip) 0.0 plus 1.0
\glue(\baselineskip) 5.16669
```

but now there is another `\parskip` glue (that is always 0pt):

```
\glue(\parskip) 0.0 plus 1.0
\glue(\parskip) 0.0
\glue(\baselineskip) 5.16669
```

The reason is that we generate a “fake” paragraph to gain control and safely add the early hooks, but this generates an additional glue item. That item doesn't contribute anything vertically but if somebody writes code that unravels a constructed list using `\lastbox`, `\unskip` and `\unpenalty` then the code has to remove one additional glue item or else it will fail.

---

<sup>26</sup>Note that this can alter the document pagination, because a paragraph ending in a display (e.g., an equation) will get an extra line—in that case our tiny number has an effect even though it doesn't take up any space, because it paragraph is no longer empty and thus isn't dropped!

### 3 The Implementation

```
1 <@=para>
2 <*2ekernel | 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 of definition for para/before and others. These functions are documented on page 344.)

\@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 of definition for \@kernel@before@para@before and others. These functions are documented on page 345.)

\g\_para\_standard\_everypar\_t1

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 <latexrelease> \IncludeInRelease{2023/06/01}
13 <latexrelease> {\g_para_standard_everypar_t1}{minipage~ fix}
14 \tl_new:N \g_para_standard_everypar_t1
```

Here is now its definition:

```
15 \tl_gset:Nn \g_para_standard_everypar_t1 {
```

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.

```
16 \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.

```
17 \group_begin:
18 \tex_par:D
19 \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.

```
20 \tex_everypar:D { \msg_error:n { hooks }{ para-mode }{before}{vertical} }
21 \@kernel@before@para@before
22 \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.

```
23 \group_begin:
24 \tex_everypar:D {}
```

There has been a long-standing problem with L<sup>A</sup>T<sub>E</sub>X's minipages in that invisible material at the beginning of a minipage (such as a `\color` setting) would result in `\parskip` being added in front of the first paragraph—something that is not done by T<sub>E</sub>X if a vertical list is completely empty. As this is happening on a very low-level in the engine it wasn't really possible to find out if this `\parskip` was added or if a space we see in front of the current point is legitimate. However, with the new paragraph handling we are in a better position: while we still don't know if there is such a space or not, we do know if we have just created an empty paragraph. Thus, if we now set `\parskip` to `-\parskip` the two will cancel each other if present and if the first was ignored because the vertical list was empty, then the second will be ignored too because it is still empty. Of course, we don't want to cancel always but only at the start of a minipage and that is signaled with the `@minipage` switch.

```
25 \skip_set:Nn \tex_parskip:D
26 { \if@minipage -\tex_parskip:D \else: \c_zero_skip \fi: }
27 \tex_noindent:D
28 \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.

```
29 \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.

```
30 \@kernel@before@para@begin
31 \hook_use:n {\para/begin}
```

If we aren't in horizontal mode any longer the hooks above misbehaved.

```
32 \if_mode_horizontal: \else:
33 \msg_error:nnnn { hooks }{ para-mode }{begin}{vertical} \fi:
```

Finally we reinsert the indentation box (unless suppressed) and then call `\everypar` the way legacy L<sup>A</sup>T<sub>E</sub>X code expects it.

However, adding the public `\everypar` is a bit tricky (see below) so we add that later, and indirectly.

```
34 __para_handle_indent:
35 % \the \everypar % <--- done differently below
36 }
37 \end{macro}
38 \end{macro}
39 \end{macro}
40 \end{macro}
41 \end{macro}
42 \end{macro}
43 \end{macro}
44 \end{macro}
45 \end{macro}
```

```

46 <latexrelease> \tex_everypar:D { \msg_error:nnn { hooks }{ para-mode }{before}{vertical} }
47 <latexrelease> @kernel@before@para@before
48 <latexrelease> \hook_use:n {para/before}
49 <latexrelease> \group_begin:
50 <latexrelease> \tex_everypar:D {}
51 <latexrelease> \skip_zero:N \tex_parskip:D
52 <latexrelease> \tex_noindent:D
53 <latexrelease> \group_end:
54 <latexrelease> \tex_everypar:D{\g__para_standard_everypar_tl}
55 <latexrelease> @kernel@before@para@begin
56 <latexrelease> \hook_use:n {para/begin}
57 <latexrelease> \if_mode_horizontal: \else:
58 <latexrelease> \msg_error:nnn { hooks }{ para-mode }{begin}{vertical} \fi:
59 <latexrelease> __para_handle_indent:
60 <latexrelease>

```

We also have to add the `\everypar` toks register at the end. In case of rollback this is already allocated and we have to find out the correct number (hope this is correctly done)

```

61 <latexrelease> \cs_set:Npn __para_tmp:w #1#2#3#4#5 { }
62 <latexrelease> \tl_gput_right:Nx \g__para_standard_everypar_tl {
63 <latexrelease> \exp_not:N \the
64 <latexrelease> \exp_not:N \toks
65 <latexrelease> \exp_after:wN __para_tmp:w \token_to_meaning:N \everypar
66 <latexrelease> \c_space_tl
67 <latexrelease>
68 <latexrelease> \EndIncludeInRelease

```

(End of definition for `\g__para_standard_everypar_tl`.)

`\tex_everypar:D` then only has to execute `\g__para_standard_everypar_tl` by default.

```
69 \tex_everypar:D{\g__para_standard_everypar_tl}
```

(End of definition for `\tex_everypar:D`.)

`\everypar` Tokens inserted at the beginning of the paragraph are placed into `\everypar` inside legacy L<sup>A</sup>T<sub>E</sub>X code, e.g., by the list environments or by headings to handle `\clubpenalty`, etc. Now this isn't any longer the primitive but simply a toks register used in the code above but to legacy L<sup>A</sup>T<sub>E</sub>X code that is transparent.

There is, however, a problem: a handful packages use exactly the same trick and replace the primitive with a token register and call the token register inside the renamed primitive. That is they assume that `\everypar` is the primitive and that it will still be called at the start of the paragraph even if renamed.

But if we have already replaced it by a token register then all they do is to give that token register a new name. Thus our code in `\tex_everypar:D` would call `\everypar` (which is now their token register) and the code that they added ends up in our token register which is then never used at all. A bit mind boggling I guess.

So what we have to do is not to call the token register `\everypar` by its name inside `\tex_everypar:D` but by using its actual register number.

```
70 \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.

```

71 \tl_gput_right:Nx \g__para_standard_everypar_tl {
72 \exp_not:N \the
73 \exp_not:N \toks
74 \the \allocationnumber
75 \c_space_tl
76 }
```

(End of definition for `\everypar`.)

**\g\_para\_indent\_box** For managing the indentation we need to provide a public accessible box register

```
77 \box_new:N \g_para_indent_box
```

(End of definition for `\g_para_indent_box`. This function is documented on page 346.)

**\\_\_para\_handle\_indent:** Adding (typesetting) the indent box is straight forward. If it was emptied before it does nothing.

```

78 \cs_new:Npn __para_handle_indent: {
79 \box_use_drop:N \g_para_indent_box
80 }
```

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

```

81 \cs_new:Npn \paraomit_indent: {
82 \box_gclear:N \g_para_indent_box
83 }
```

(End of definition for `\__para_handle_indent`.)

**\IndentBox** **\OmitIndent** The L<sup>A</sup>T<sub>E</sub>X2<sub><</sub> names for the indentation box and for suppressing it for use in the para/begin hook.

```

84 \cs_set_eq:NN \IndentBox \g_para_indent_box
85 \cs_set_eq:NN \OmitIndent \paraomit_indent:
```

(End of definition for `\IndentBox` and `\OmitIndent`. These functions are documented on page 346.)

**\para\_end:** Adding hooks to the end of a paragraph is similar but here we need to alter the command that is used by T<sub>E</sub>X to end horizontal mode and return to vertical mode, i.e., `\par`.

This is a bit more complicated as this command can appear anywhere either explicitly or implicitly added by T<sub>E</sub>X in certain situations:

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

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

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

We are now going to change this default meaning to instead run  $\text{\para_end:}$ , which ultimately executes the  $\text{initex}$  primitive but additionally adds our hooks when appropriate. This way the change is again transparent to the legacy  $\text{\LaTeX} 2_{\varepsilon}$  code.

In most cases  $\text{\para_end:}$  should behave exactly like the primitive and we achieve this by simply expanding it to the primitive which is available to us as  $\text{\tex_par:D}$ . This way we don't have to care about whether  $\text{\TeX}$  just does nothing (e.g., if in vertical mode already) or generates an error, etc.

```
86 \cs_new_protected:Npn \para_end: {
```

CCC Maybe needs more explanation. TEMP NOTE: What should happen if in outer hmode with an empty hlist?

The only case we care about is when we are in horizontal mode (i.e., doing typesetting) and not also in inner mode (i.e., making paragraphs and not building an  $\text{\hbox}$ ).

```
\bool_lazy_and:nnT
 { \mode_if_horizontal_p: }
 { \bool_not_p:n { \mode_if_inner_p: } }
 { ... }
```

Since this is executed for each and every paragraph in a document we try to stay as fast as possible, so we do not use the above construct but two conditionals instead. Using low-level  $\text{\if_mode...}$  conditions would be even faster but has the danger to conflict with conditionals in the user hooks.

If  $\text{\para_end:}$  is executed while  $\text{\TeX}$  is currently doing a low-level assignment the test for horizontal mode may get executed as part of the assignment. That is normally not an issue but we just found one case where it is:

```
\afterassignment\lst@vskip\@tempskipa \z@\par
```

If  $\text{\TeX}$  is in hmode while that assignment happens then the  $\text{\par}$  is seen in hmode because in the above case the assignment may not be finished (one should have used  $\text{\z@skip}$ ) and the  $\text{\lst@vskip}$  will get inserted into the middle of the conditional. The  $\text{\lst@vskip}$  then changes to vmode and you get a surprising error about the  $\text{para/end}$  hook having changed modes even if you don't have any hook code(!): it is the inserted  $\text{\lst@vskip}$  that is actually causing the change of mode. This is what happened when the output routines got started while a  $\text{lstlisting}$  environment (that redefines  $\text{\vskip}$  in this way) was active. This is really faulty coding, but we try to be proactive and guard the conditional so that any scanning is first stopped, thus:

```
87 \scan_stop:
88 \mode_if_horizontal:TF {
89 \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.

```
90 \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.

```
91 \hook_use:n{para/end}
92 \@kernel@after@para@end
93 \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_lastnodeltype: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`.

```
94 \if_int_compare:w 11 = \tex_lastnodeltype:D
95 \tex_hskip:D \c_zero_dim
96 \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.

```

97 \tex_par:D
98 \hook_use:n{para/after}
99 \kernel@after@para@after
100 }

```

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.

```
101 { \msg_error:nnn { hooks }{ para-mode }{end}{horizontal} }
```

Finally close out the nested conditionals.

```

102 }
103 }
```

And then we can use the primitive to truly end the paragraph.

```

104 \tex_par:D
105 }
```

*(End of definition for `\para_end`. This function is documented on page 345.)*

`\para_raw_indent:`

`\para_raw_noindent:`

`\para_raw_end:`

The commands `\para_raw_indent:` and `\para_raw_noindent:` are like the primitives `\indent` and `\noindent` except that they can only be used in vertical mode.

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.

```

106 \cs_new:Npn \para_raw_indent: {
107 \mode_if_vertical:TF
108 {
109 \tex_everypar:D {
110 \box_gset_to_last:N \g_para_indent_box
111 \tex_everypar:D { \g__para_standard_everypar_tl }
112 __para_handle_indent:
113 \the\tex_everypar }
114 }
115 { \msg_error:nn { latex2e }{ raw-para } }
116 \tex_indent:D
117 }

118 \cs_new:Npn \para_raw_noindent: {
119 \mode_if_vertical:TF
120 {
121 \tex_everypar:D {
122 \tex_everypar:D { \g__para_standard_everypar_tl }
123 \the\tex_everypar }
124 }
125 { \msg_error:nn { latex2e }{ raw-para } }
126 \tex_noindent:D
127 }

128 \cs_new_eq:NN \para_raw_end: \tex_par:D

```

*(End of definition for `\para_raw_indent`, `\para_raw_noindent`, and `\para_raw_end`. These functions are documented on page 346.)*

```

\RawIndent The LATEX 2< names for starting and ending a paragraph without adding any hooks.
\RawNoIndent 129 \cs_set_eq:NN \RawIndent \para_raw_indent:
\RawParEnd 130 \cs_set_eq:NN \RawNoindent \para_raw_noindent:
131 \cs_set_eq:NN \RawParEnd \para_raw_end:

```

(End of definition for \RawIndent, \RawNoIndent, and \RawParEnd. These functions are documented on page 346.)

This ends the para module code.

```
132 <@=
```

\par Having the new default definition for \par we also have to set it up so that it gets used. This involves three commands: \par, \@@par (to which L<sup>A</sup>T<sub>E</sub>X resets \par occasionally) and \endgraf, which is another name for the “default” action of \par.

```

133 \cs_set_eq:NN \par \para_end:
134 \cs_set_eq:NN \@@par \para_end:
135 \cs_set_eq:NN \endgraf \para_end:

```

(End of definition for \par, \endgraf, and \@@par. These functions are documented on page 345.)

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).

```
136 \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.

```

137 \msg_new:nnnn { hooks } { para-mode }
138 {
139 Illegal~mode~ change~ in~ hook~ 'para/#1'.\\
140 Hook~ code~ did~ not~ remain~ in~ #2~ mode.
141 }
142 {
143 Paragraph~ hooks~ cannot~ change~ the~ TeX~ mode~ without~ causing~
144 endless~ recursion.~ The~ hook~ code~ in~ 'para/#1'~ needs~ to~ stay~
145 in~ #2~ mode,~ but~ it~ didn't.~ Examine~ the~ hook~
146 code~ with~ \iow_char:N \\ShowHook~ to~ find~ the~ issue.
147 }

```

And here is one used in the “raw” commands when they are used outside of vertical mode.

```

148 \msg_new:nnnn { latex2e } { raw-para }
149 {
150 Not~ in~ vertical~ mode.
151 }
152 {
153 Starting~ a~ paragraph~ with~ \iow_char:N \\RawIndent~ or~
154 \iow_char:N \\RawNoindent \\
155 (or~ \iow_char:N \\para_raw_indent:~ or~
156 \iow_char:N \\para_raw_noindent:)~ is~ only~ allowed \\
157 if~ LATEX~ is~ in~ vertical~ mode.
158 }

```

```

159 %
160 <|latexrelease>\IncludeInRelease{0000/00/00}%
161 <|latexrelease> {ltpara}{Undo-hooks-for-paragraphs}
162 <|latexrelease>
163 <|latexrelease>\let \OmitIndent \@undefined
164 <|latexrelease>\let \IndentBox \@undefined
165 <|latexrelease>\let \RawIndent \@undefined
166 <|latexrelease>\let \RawNoindent \@undefined
167 <|latexrelease>\let \RawParEnd \@undefined
168 <|latexrelease>
169 <|latexrelease>\cs_set_eq:NN \par \tex_par:D
170 <|latexrelease>\cs_set_eq:NN \@@par \tex_par:D
171 <|latexrelease>\cs_set_eq:NN \endgraf \tex_par:D
172 <|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.

```

173 <|latexrelease>\tex_everypar:D {}
174 <|latexrelease>\cs_set_eq:NN \everypar \tex_everypar:D
175 <|latexrelease>
176 <|latexrelease>\EndModuleRelease
177 \ExplSyntaxOff
178 </2ekernel | latexrelease>

```

# File o

## ltmeta.dtx

### Abstract

This code defines the `\DocumentMetadata` interface.

## 1 Introduction

In the past there was no dedicated location to declare settings concerning a document as a whole. Settings are placed somewhere in the preamble or with the class options or even with some package options. For some settings this can be too late, for example the pdf version can no longer be changed if a package has used code which already opened the PDF.

`\DocumentMetadata` as a new command unifies such settings in one place. It must be used before `\documentclass` but can be issued more than once there.

At the moment most of the code run by `\DocumentMetadata` is external to the format and subject to change. This includes the supported key/values.

For that reason all that happens right now in the format is to look for suitable support files and if found, to redirect the processing to them.

### 1.1 `\DocumentMetadata`

---

```
\DocumentMetadata \DocumentMetadata{\{key-value list\}}
```

---

The keys defined for `\DocumentMetadata` currently allow to set the PDF version, to set the PDF `/Lang`, to uncompress a PDF, to set the language and to declare a few PDF standards and some color profiles.

`\DocumentMetadata` is also used to activate the new PDF management code and it loads a number of required files for the PDF management code. As this forces the loading of the backend files, a backend which can't be detected automatically like `dvipdfmx`, must be set in the first `\DocumentMetadata` call (if there is more than one).

The full set of keys currently supported is documented in `documentmetadata-support.pdf` for now.

## 2 The Implementation

```
1 {*2ekernel | latexrelease}
Not needed yet but ...
2 %\ExplSyntaxOn
3 \let \IfDocumentMetadataTF \@secondoftwo
4 \protected\def\DocumentMetadata{%
5 \InputIfFileExists{documentmetadata-support.ltx}%
6 {}%
7 }%
```

The above file is changing `\DocumentMetadata` to a suitable definition (or so we hope), so that we can try again — if not tough.

If the file can't be found we say so and carry on without it.

```

9 {%
10 \@latex@error{No support files for
11 \noexpand\DocumentMetadata found}
12 {Is the 'LaTeX-lab' bundle installed?%
13 \MessageBreak
14 Without it, the declaration is ignored.}%

```

No point in trying this more than once if there are several calls in the document.

```

15 \let\DocumentMetadata\@gobble
16 }%
17 \let \IfDocumentMetadataTF \@firstoftwo
18 \DocumentMetadata
19 }

```

To allow package and class author to support for document links we provide also the new interface commands of the hyperref package for the creation of targets.

```

\MakeLinkTarget
 \LinkTargetOn
 \LinkTargetOff
\NextLinkTarget
20 \NewDocumentCommand\MakeLinkTarget{s0{}m}{%
21 \ifvmode
22 \special{}%
23 \else
24 \@savsf\spacefactor
25 \smash{}%
26 \spacefactor@savsf
27 \fi}
28 \NewDocumentCommand\LinkTargetOn{}{}
29 \NewDocumentCommand\LinkTargetOff{}{}
30 \NewDocumentCommand\NextLinkTarget{m}{}

```

(End of definition for `\MakeLinkTarget` and others.)

We do not undo `\MakeLinkTarget` and friends if we roll back, in case they are used in packages that themselves do not offer rollback. This way a roll forward adds them, but the dummies remain if you roll back and you don't get missing csname errors if they are used.

```

31 <| latexrelease> \IncludeInRelease{0000/00/00}{ltmeta}%
32 <| latexrelease> {Undo Document Metadata handling}
33 <| latexrelease>
34 <| latexrelease> \let\DocumentMetadata\@undefined
35 <| latexrelease>
36 <| latexrelease> \EndModuleRelease

```

Again for the future ...

```

37 %\ExplSyntaxOff
38 </2ekernel | latexrelease>

```

Restore module prefix (if any):

```

39 <@@=
```

# File p

## ltspacex.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
 \predisplaypenalty
 \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):
 \obeyscr : defines <CR> == \\relax
 \restorescr : 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 of definition for \pagebreak and \nopagebreak.)

\linebreak
\nolinebreak
9 \DeclareRobustCommand{\linebreak}{\testopt{\no@lnbk-}4}
10 \DeclareRobustCommand{\nolinebreak}{\testopt\no@lnbk4}

(End of definition for \linebreak and \nolinebreak.)

\samepage
11 \DeclareRobustCommand{\samepage}{\interlinepenalty\OM
12 \predisplaypenalty\OM
13 \postdisplaypenalty\OM
14 \interdisplaylinepenalty\OM
15 \beginparpenalty\OM
16 \endparpenalty\OM
17 \itempenalty\OM
18 \secpenalty\OM
19 \interfootnotelinepenalty\OM}

(End of definition for \samepage.)

20 {/2ekernel | latexrelease}
21 {latexrelease} \EndIncludeInRelease
22 {latexrelease} \IncludeInRelease{0000/00/00}%
23 {latexrelease} {\pagebreak}{Make commands robust}%
24 {latexrelease}
25 {latexrelease} \kernel@make@fragile\pagebreak
26 {latexrelease} \kernel@make@fragile\nopagebreak
27 {latexrelease} \kernel@make@fragile\linebreak
28 {latexrelease} \kernel@make@fragile\nolinebreak
29 {latexrelease} \kernel@make@fragile\samepage
30 {latexrelease}
31 {latexrelease} \EndIncludeInRelease
32 {*2ekernel}

```

```

\@no@pgbk
33 \def\@no@pgbk #1[#2]{%
34 \ifvmode
35 \penalty #1\@getpen{#2}%
36 \else
37 \@bsphack
38 \vadjust{\penalty #1\@getpen{#2}}%
39 \@esphack
40 \fi}

```

(End of definition for \@no@pgbk.)

```

\@no@lnbk
41 \def\@no@lnbk #1[#2]{%
42 \ifvmode
43 \nolnerr
44 \else
45 \tempskipa\lastskip
46 \unskip
47 \penalty #1\@getpen{#2}%
48 \ifdim\tempskipa>\z@
49 \hskip\tempskipa
50 \ignorespaces
51 \fi
52 \fi}

```

(End of 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
53 </2ekernel>
54 <*2ekernel | latexrelease>
55 <latexrelease> \IncludeInRelease{2020/02/02}%
56 <latexrelease> {\@normalcr}{Make robust}%
57 \protected\def\@normalcr{%
58 \let \reserved@e \relax
59 \let \reserved@f \relax
60 \@ifstar{\let \reserved@e \vadjust \let \reserved@f \nobreak
61 \xnewline}%
62 \xnewline}

```

```

63 \let\\@normalcr
64 </2ekernel | latexrelease>
65 <latexrelease>\EndIncludeInRelease
66 <latexrelease>\IncludeInRelease{0000/00/00}%
67 <latexrelease> {\@normalcr}{Make robust}%
68 <latexrelease>
69 <latexrelease>\DeclareRobustCommand\\{%
70 <latexrelease> \let \reserved@e \relax
71 <latexrelease> \let \reserved@f \relax
72 <latexrelease> \@ifstar{\let \reserved@e \vadjust \let \reserved@f \nobreak
73 <latexrelease> \xnewline}%
74 <latexrelease> \xnewline}
75 <latexrelease>\expandafter\let\expandafter\@normalcr
76 <latexrelease> \csname\expandafter\gobble\string\\ \endcsname
77 <latexrelease>
78 <latexrelease>\EndIncludeInRelease
79 <*2ekernel>

```

(End of 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.

```

80 </2ekernel>
81 <*2ekernel | latexrelease>
82 <latexrelease>\IncludeInRelease{2020/10/01}%
83 <latexrelease> {\@vspace@calcify}{Add calc support}%
84 \def\@vspace@calcify#1{\setlength\sp@ce@skip{#1}\vskip\sp@ce@skip}
85 </2ekernel | latexrelease>
86 <latexrelease>\EndIncludeInRelease
87 <latexrelease>\IncludeInRelease{0000/00/00}%
88 <latexrelease> {\@vspace@calcify}{Add calc support}%
89 <latexrelease>
90 <latexrelease>\let\@vspace@calcify\@undefined
91 <latexrelease>\EndIncludeInRelease
92 <*2ekernel>

```

(End of definition for \@vspace@calcify.)

**\newline** A simple form of the ‘normal’ definition of \\.

```

93 \DeclareRobustCommand\newline{\@normalcr\relax}

```

(End of definition for \newline.)

**\@xnewline**

```

94 \def\@xnewline{\@ifnextchar[%] bracket matching
95 \newline
96 {\@gnewline\relax}}

```

(End of definition for \@xnewline.)

**\@newline**

```

97 </2ekernel>
98 <*2ekernel | latexrelease>
99 <latexrelease>\IncludeInRelease{2020/10/01}%

```

```

100 <{latexrelease} {\@newline}{\newline calc support}%
101 \def\@newline[#1]{\let \reserved@e \vadjust
102 \@gnewline {\@vspace@calcify{#1}}}
103 </2ekernel | latexrelease>
104 <{latexrelease}\EndIncludeInRelease
105 <{latexrelease}\IncludeInRelease{0000/00/00}%
106 <{latexrelease} {\@newline}{\newline calc support}%
107 <{latexrelease}>
108 <{latexrelease}\def\@newline[#1]{\let \reserved@e \vadjust
109 <{latexrelease} \@gnewline {\vskip #1}}
110 <{latexrelease}\EndIncludeInRelease
111 <{*2ekernel}>

```

(End of 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

```

112 \def\@gnewline #1{%
113 \ifvmode
114 \nolnerr
115 \else
116 \unskip \reserved@e {\reserved@f#1}\nobreak \hfil \break
117 \fi}

```

(End of definition for `\@gnewline`.)

`\@getpen`

```

118 \def\@getpen#1{\ifcase #1 \z@ \or \clowpenalty\or
119 \medpenalty \or \chighpenalty
120 \else \z@\fi}

```

(End of 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.

```

121 \def\@nobreakfalse{\global\let\if@nobreak\iffalse}
122 \def\@nobreaktrue {\global\let\if@nobreak\iftrue}
123 \z@\nobreakfalse

```

(End of definition for `\if@nobreak`.)

`\@savsk` Registers used to save the space factor and last skip.

```

124 \newdimen\@savsk
125 \newcount\@savsf

```

(End of 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.

```

126 \def\@bsphack{%
127 \relax
128 \ifhmode
129 \csavsk\lastskip
130 \csavsf\spacefactor
131 \fi}

```

(End of 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.

```

132 </2ekernel>
133 <latexrelease>\IncludeInRelease{2018/10/10}%
134 <latexrelease> {\@esphack}{hyphenation and nobreak after space hack}%
135 <*2ekernel | latexrelease>
136 \def\@esphack{%
137 \relax
138 \ifhmode
139 \spacefactor\@savsf
140 \ifdim\@savsk>\z@
141 \ifdim\lastskip=\z@
142 \nobreak \hskip\z@skip
143 \fi
144 \ignorespaces
145 \fi
146 \else
147 \ifvmode
148 \if@nobreak\nobreak\else\if@noskipsec\nobreak\fi\fi
149 \fi
150 \fi}%
151 </2ekernel | latexrelease>
152 <latexrelease>\EndIncludeInRelease
153 <latexrelease>\IncludeInRelease{2015/10/01}%
154 <latexrelease> {\@esphack}{hyphenation and nobreak after space hack}%
155 <latexrelease>\def\@esphack{%
156 <latexrelease> \relax
157 <latexrelease> \ifhmode
158 <latexrelease> \spacefactor\@savsf
159 <latexrelease> \ifdim\@savsk>\z@
160 <latexrelease> \ifdim\lastskip=\z@
161 <latexrelease> \nobreak \hskip\z@skip
162 <latexrelease> \fi

```

```

163 <{latexrelease}> \ignorespaces
164 <{latexrelease}> \fi
165 <{latexrelease}> \fi}%
166 <{latexrelease}>\EndIncludeInRelease
167 <{latexrelease}>\IncludeInRelease{2015/01/01}%
168 <{latexrelease}> {\@esphack}{hyphenation and nobreak after space hack}%
169 <{latexrelease}>\def\@esphack{%
170 <{latexrelease}> \relax
171 <{latexrelease}> \ifhmode
172 <{latexrelease}> \spacefactor\@savsf
173 <{latexrelease}> \ifdim\@savsk>\z@
174 <{latexrelease}> \nobreak \hskip\z@skip
175 <{latexrelease}> \ignorespaces
176 <{latexrelease}> \fi
177 <{latexrelease}> \fi}%
178 <{latexrelease}>\EndIncludeInRelease
179 <{latexrelease}>\IncludeInRelease{0000/00/00}%
180 <{latexrelease}> {\@esphack}{hyphenation and nobreak after space hack}%
181 <{latexrelease}>\def\@esphack{%
182 <{latexrelease}> \relax
183 <{latexrelease}> \ifhmode
184 <{latexrelease}> \spacefactor\@savsf
185 <{latexrelease}> \ifdim\@savsk>\z@
186 <{latexrelease}> \ignorespaces
187 <{latexrelease}> \fi
188 <{latexrelease}> \fi}%
189 <{latexrelease}>\EndIncludeInRelease
190 <{*2ekernel}>

```

(End of 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

```

191 </2ekernel>
192 <{latexrelease}>\IncludeInRelease{2015/01/01}%
193 <{latexrelease}> {\@Eshack}{hyphenation after space hack}%
194 <{*2ekernel | latexrelease}>
195 \def\@Eshack{%
196 \relax
197 \ifhmode
198 \spacefactor\@savsf
199 \ifdim\@savsk>\z@
200 \nobreak \hskip\z@skip
201 \@ignoretrue
202 \ignorespaces
203 \fi
204 \fi}%
205 </2ekernel | latexrelease>
206 <{latexrelease}>\EndIncludeInRelease
207 <{latexrelease}>\IncludeInRelease{0000/00/00}%
208 <{latexrelease}> {\@Eshack}{hyphenation after space hack}%
209 <{latexrelease}>\def\@Eshack{%
210 <{latexrelease}> \relax
211 <{latexrelease}> \ifhmode

```

```

212 〈\latexrelease〉 \spacefactor\@savsf
213 〈\latexrelease〉 \ifdim\@savsk>\z@
214 〈\latexrelease〉 \ignorespacestrue
215 〈\latexrelease〉 \ignorespaces
216 〈\latexrelease〉 \fi
217 〈\latexrelease〉 \fi}%
218 〈\latexrelease〉\EndIncludeInRelease
219 {*2ekernel}

```

(End of 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 of definition for \@vbsphack.)

## 1.5 Vertical spacing

L<sup>A</sup>T<sub>E</sub>X supports the plain T<sub>E</sub>X commands \smallskip, \medskip and \bigskip. However, it redefines them using \vspace instead of \skip.

Extra vertical space is added by the command \addvspace{\<skip>}, which adds a vertical skip of <skip> to the document. The sequence \addvspace{\<s1>} \addvspace{\<s2>} is equivalent to \addvspace{\<maximum of s1, s2>}.

\addvspace should be used only in vertical mode, and gives an error if it's not. The \addvspace command does *not* add vertical space if @minipage is true. The minipage environment uses this to inhibit the addition of extra vertical space at the beginning.

Penalties are put into the vertical list with the \addpenalty{\<penalty>} command. It works properly when \addpenalty and \addvspace commands are mixed.

The @nobreak switch is set true used when in vertical mode and no page break should occur. (Right now, it is used only by the section heading commands to inhibit page breaking after a heading.)

```

\addvspace{SKIP} ==
BEGIN
 if vmode
 then if @minipage

```

```

 else if \lastskip =0
 then \vskip SKIP
 else if \lastskip < SKIP
 then \vskip -\lastskip
 \vskip SKIP
 else if SKIP < 0 and \lastskip >= 0
 then \vskip -\lastskip
 \vskip \lastskip + SKIP
 fi fi fi fi
 else useful error message (CAR).
fi
END

```

\@xaddvskip Internal macro for \vspace handling the case that space has previously been added.

```

220 \def\@xaddvskip{%
221 \ifdim\lastskip<\@tempskipb
222 \vskip-\lastskip
223 \vskip\@tempskipb
224 \else
225 \ifdim\@tempskipb<\z@
226 \ifdim\lastskip<\z@
227 \else
228 \advance\@tempskipb\lastskip
229 \vskip-\lastskip
230 \vskip \@tempskipb
231 \fi
232 \fi
233 \fi}

```

*(End of definition for \@xaddvskip.)*

\addvspace Add vertical space taking into account space already added, as described above.

```

234 </2ekernel>
235 <*2ekernel | latexrelease>
236 <latexrelease>\IncludeInRelease{2020/10/01}%
237 <latexrelease> {\addvspace}{\addvspace calc support}%
238 \def\addvspace#1{%
239 \ifvmode
240 \if@minipage\else
241 \ifdim \lastskip =\z@
242 \vspace@calcify{#1}%
243 \else
244 \setlength\@tempskipb{#1}%
245 \vskip\@xaddvskip
246 \fi
247 \fi
248 \else
249 \noitemerr
250 \fi}
251 </2ekernel | latexrelease>
252 <latexrelease>\EndIncludeInRelease
253 <latexrelease>\IncludeInRelease{0000/00/00}%
254 <latexrelease> {\addvspace}{\addvspace calc support}%

```

```

255 〈\latexrelease〉
256 〈\latexrelease〉\def\addvspace#1{%
257 〈\latexrelease〉 \ifvmode
258 〈\latexrelease〉 \if@minipage\else
259 〈\latexrelease〉 \ifdim \lastskip =\z@
260 〈\latexrelease〉 \vskip #1\relax
261 〈\latexrelease〉 \else
262 〈\latexrelease〉 \tempskipb#1\relax
263 〈\latexrelease〉 \xaddvskip
264 〈\latexrelease〉 \fi
265 〈\latexrelease〉 \fi
266 〈\latexrelease〉 \else
267 〈\latexrelease〉 \noitemerr
268 〈\latexrelease〉 \fi}
269 〈\latexrelease〉\EndIncludeInRelease
270 〈*2ekernel〉

```

(End of definition for `\addvspace`.)

### `\addpenalty`

```

271 〈/2ekernel〉
272 〈\latexrelease〉\IncludeInRelease{2015/01/01}%
273 〈\latexrelease〉 {\addpenalty}{\addpenalty}%
274 〈*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.

```

275 \def\addpenalty#1{%
276 \ifvmode
277 \if@minipage
278 \else
279 \if@nobreak
280 \else
281 \ifdim\lastskip=\z@
282 \penalty#1\relax
283 \else
284 \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.

```

285 \begingroup
286 \tempskipa\tempskipb
287 \advance \tempskipb
288 \ifdim\prevdepth>\maxdepth\maxdepth\else

```

If `\prevdepth` is -1000pt due to `\nointerlineskip` we better not add it!

```

289 \ifdim \prevdepth = -\@m\p@ \z@ \else \prevdepth \fi
290 \fi
291 \vskip -\tempskipb
292 \penalty#1%
293 \ifdim\tempskipa=\tempskipb

```

Do nothing if the `\prevdepth` check made no adjustment.

294                   `\else`

Combine the prevdepth adjustment into a single skip.

295                   `\advance\@tempskipb -\@tempskipa`  
296                   `\vskip \@tempskipb`  
297                   `\fi`

The final skip is always the specified length.

298                   `\vskip \@tempskipa`  
299                   `\endgroup`  
300                   `\fi`  
301                   `\fi`  
302                   `\fi`  
303                   `\else`  
304                   `\@noitemerr`  
305                   `\fi} %`  
  
306                   `//2ekernel | latexrelease)`  
307                   `\langle latexrelease\rangle\EndIncludeInRelease`  
308                   `\langle latexrelease\rangle\IncludeInRelease{0000/00/00}%`  
309                   `\langle latexrelease\rangle \{\addpenalty\}{\addpenalty}\%`  
310                   `\langle latexrelease\rangle\def\addpenalty#1{%`  
311                   `\langle latexrelease\rangle \ifvmode`  
312                   `\langle latexrelease\rangle \if@minipage`  
313                   `\langle latexrelease\rangle \else`  
314                   `\langle latexrelease\rangle \if@nobreak`  
315                   `\langle latexrelease\rangle \else`  
316                   `\langle latexrelease\rangle \ifdim\lastskip=\z@`  
317                   `\langle latexrelease\rangle \penalty#1\relax`  
318                   `\langle latexrelease\rangle \else`  
319                   `\langle latexrelease\rangle \@tempskipb\lastskip`  
320                   `\langle latexrelease\rangle \vskip -\lastskip`  
321                   `\langle latexrelease\rangle \penalty#1%`  
322                   `\langle latexrelease\rangle \vskip\@tempskipb`  
323                   `\langle latexrelease\rangle \fi`  
324                   `\langle latexrelease\rangle \fi`  
325                   `\langle latexrelease\rangle \fi`  
326                   `\langle latexrelease\rangle \else`  
327                   `\langle latexrelease\rangle \@noitemerr`  
328                   `\langle latexrelease\rangle \fi} %`  
329                   `\langle latexrelease\rangle\EndIncludeInRelease`  
330                   `(*2ekernel)`

(End of 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.

331                   `\DeclareRobustCommand\vspace{\@ifstar\@vspacer\@vspace}`

```

332 </2ekernel>
333 <*2ekernel | latexrelease>
334 <latexrelease>\IncludeInRelease{2020/10/01}%
335 <latexrelease> {\@vspace}{Support calc in \vspace}%

We support calc syntax in the argument and therefore use \setlength.

336 \def\@vspace #1{%
337 \ifvmode
338 \@vspace@calcify{#1}%
339 \vskip\z@skip
340 \else
341 \@bsphack
342 \vadjust{\@restorepar
343 \@vspace@calcify{#1}%
344 \vskip\z@skip
345 }%
346 \@esphack
347 \fi}

348 \def\@vspacer#1{%
349 \ifvmode
350 \dimen@\prevdepth
351 \hrule\@height\z@
352 \nobreak
353 \@vspace@calcify{#1}%
354 \vskip\z@skip
355 \prevdepth\dimen@
356 \else
357 \@bsphack
358 \vadjust{\@restorepar
359 \hrule\@height\z@
360 \nobreak
361 \@vspace@calcify{#1}%
362 \vskip\z@skip}%
363 \@esphack
364 \fi}
365 </2ekernel | latexrelease>
366 <latexrelease>\EndIncludeInRelease
367 <latexrelease>\IncludeInRelease{0000/00/00}%
368 <latexrelease> {\@vspace}{Support calc in \vspace}%
369 <latexrelease>
370 <latexrelease>\def\@vspace #1{%
371 \ifvmode
372 \vskip #1
373 \vskip\z@skip
374 \else
375 \@bsphack
376 \vadjust{\@restorepar
377 \vskip #1
378 \vskip\z@skip
379 }%
380 \@esphack
381 \fi}
382 <latexrelease>\def\@vspacer#1{%
383 \ifvmode

```

```

384 ⟨latexrelease⟩ \dimen@ \prevdepth
385 ⟨latexrelease⟩ \hrule \height \z@
386 ⟨latexrelease⟩ \nobreak
387 ⟨latexrelease⟩ \vskip #1
388 ⟨latexrelease⟩ \vskip \z@skip
389 ⟨latexrelease⟩ \prevdepth \dimen@
390 ⟨latexrelease⟩ \else
391 ⟨latexrelease⟩ \@bsphack
392 ⟨latexrelease⟩ \vadjust{\@restorepar
393 ⟨latexrelease⟩ \hrule \height \z@
394 ⟨latexrelease⟩ \nobreak
395 ⟨latexrelease⟩ \vskip #1
396 ⟨latexrelease⟩ \vskip \z@skip}%
397 ⟨latexrelease⟩ \@esphack
398 ⟨latexrelease⟩ \fi}
399 ⟨latexrelease⟩ \EndIncludeInRelease
400 ⟨*2ekernel⟩

```

(End of definition for `\vspace`, `\@vspace`, and `\@vspace@`.)

```

\smallskip
\medskip 401 \def\smallskip{\vspace\smallskipamount}
\bigskip 402 \def\medskip{\vspace\medskipamount}
403 \def\bigskip{\vspace\bigskipamount}

```

(End of definition for `\smallskip`, `\medskip`, and `\bigskip`.)

```

\smallskipamount
\medskipamount 404 \newskip\smallskipamount \smallskipamount=3pt plus 1pt minus 1pt
\bigskipamount 405 \newskip\medskipamount \medskipamount =6pt plus 2pt minus 2pt
406 \newskip\bigskipamount \bigskipamount =12pt plus 4pt minus 4pt

```

(End of 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!

```

407 \DeclareRobustCommand{\nobreakdashes}{%
408 \leavevmode
409 \toks@{}%
410 \def\reserved@a##1{\toks@\expandafter{\the\toks@-}%
411 \futurelet\@let@token \reserved@b}%
412 \def\reserved@b {\ifx\@let@token -%
413 \expandafter\reserved@a

```

```

414 \else
415 \setbox\z@ \hbox{\the\toks@\nobreak}%
416 \unhbox\z@
417 \spacefactor\sfcodes`-
418 \fi}%
419 \futurelet\@let@token \reserved@b
420 }

```

(End of definition for `\nobreakdashes`.)

- `\nobreakspace` This is a robust command that produces a horizontal space at which, in paragraph-mode, a line-break is not possible. We then define an active ~ to expand to it since this is the documented behaviour of ~. One reason for introducing this is that some 8-bit input encodings have a slot for such a space and we do not want to use active characters as the L<sup>A</sup>T<sub>E</sub>X internal commands.

The braces in the definition of ~ are needed to ensure that a following space is preserved when reading to/from internal files.

We need to keep `\@xobeysp` as it is widely used; so here it is let to the non-robust command `\nobreakspace`.

```

421 \DeclareRobustCommand{\nobreakspace}{%
422 \leavevmode\nobreak\ }
423 \catcode `~=13
424 \def~{\nobreakspace{}}
425 \expandafter\let\expandafter\@xobeysp\csname nobreakspace \endcsname

```

(End of 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.

```

426 </2ekernel>
427 <latexrelease>\IncludeInRelease{2015/01/01}%
428 <latexrelease> {\@}{Space after \@}%
429 <*2ekernel | latexrelease>
430 \def\@{\spacefactor\@m{}}%
431 </2ekernel | latexrelease>
432 <latexrelease>\EndIncludeInRelease
433 <latexrelease>\IncludeInRelease{0000/00/00}%
434 <latexrelease> {\@}{Space after \@}%
435 <latexrelease>\def\@{\spacefactor\@m{}}%
436 <latexrelease>\EndIncludeInRelease
437 <*2ekernel>

```

(End of definition for `\@`.)

`\hspace`

```

438 \DeclareRobustCommand\hspace{\@ifstar\@hspacer\@hspace}

```

(End of definition for `\hspace`.)

`\@hspace`

```

439 </2ekernel>
440 <*2ekernel | latexrelease>
441 <latexrelease>\IncludeInRelease{2020/10/01}%
442 <latexrelease> {\@hspace}{Support calc with \hspace}%

```

We use a private register to calculate the space (if `calc` is used). Previously we used a group but that results in `\everypar` etc. being executed inside the group if the `\hspace` starts a paragraph. This is a bug fix so we do not provide rollback to the incorrect intermediate version.

```

443 \newskip\sp@ce@skip
444 \def\@hspace#1{\setlength\sp@ce@skip{#1}\hskip\sp@ce@skip}
445 </2ekernel | latexrelease>
446 <latexrelease>\EndIncludeInRelease
447 <latexrelease>\IncludeInRelease{0000/00/00}%
448 <latexrelease> {\@hspace}{Support calc with \hspace}%
449
450 <latexrelease>
451 <latexrelease>\def\@hspace#1{\hskip #1\relax}
452 <latexrelease>\EndIncludeInRelease
453 <*2ekernel>

```

(End of definition for `\@hspace`.)

**\@hspacer** Extra `\hskip 0pt` 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

```

454 \def\@hspacer#1{\vrule \@width\z@\nobreak
455 \@hspace{#1}\hskip \z@skip}

```

(End of definition for `\@hspacer`.)

```
\fill
456 \newskip\fill
457 \fill = 0pt plus 1fill
```

(End of definition for `\fill`.)

```
\stretch
458 \def\stretch#1{\z@ \@plus #1fill\relax}

(End of definition for \stretch.)

459 </2ekernel>
460 <*2ekernel | latexrelease>
461 <latexrelease>\IncludeInRelease{2018/12/01}%
462 <latexrelease> {\thinspace}{Start LR-mode}%
```

```
\enspace
463 \DeclareRobustCommand\enspace{\leavevmode@ifvmode\kern.5em }
```

(End of 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).

```
464 \protected\def\leavevmode@ifvmode{\ifvmode\expandafter\indent\fi}
```

```

(End of definition for \leavevmode@ifvmode.)
```

```

465 </2ekernel | latexrelease>
466 <latexrelease>\EndIncludeInRelease
467 <latexrelease>\IncludeInRelease{0000/00/00}%
468 <latexrelease> {\thinspace}{Start LR-mode}%
469 <latexrelease>\def\thinspace{\kern .16667em }
470 <latexrelease>\def\negthinspace{\kern-.16667em }
471 <latexrelease>\def\enspace{\kern.5em }
472 <latexrelease>\let\leavevmode@ifvmode@undefined
473 <latexrelease>\EndIncludeInRelease
474 {*2ekernel}
```

```

\enskip
\quad 475 \def\enskip{\hskip.5em\relax}
\quad 476 \def\quad{\hskip1em\relax}
477 \def\quad{\hskip2em\relax}

(End of definition for \enskip, \quad, and \quad.)
```

For Unicode engines, make the Unicode soft hyphen an active character defined as \-.  
478 \ifx\Umathcode\@undefined\else
479 \catcode "AD=13
480 \def^^ad{\-}
481 \fi

\obeycr The following definitions will probably get deleted or moved to compatibility mode soon.  
\restorecr
482 {\catcode`^^M=13 \gdef\obeycr{\catcode`^^M13 \def^^M{\relax}%
483 \gobblecr}%
484 {\catcode`^^M=13 \gdef\gobblecr{\@ifnextchar
485 \gobble\ignorespaces}%
486 \gdef\restorecr{\catcode`^^M5 }}

```

(End of definition for \obeycr and \restorecr.)
```

```

487 </2ekernel>
```

# File q

## ltlogos.dtx

### 1 Logos

Various logos are defined here.

- \TeX The \TeX logo, adjusted so that a full stop after the logo counts as ending a sentence.

```
1 <*2ekernel>
2 \DeclareRobustCommand{\TeX}{\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX\@}
```

(End of 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 of definition for \LaTeX.)

- \LaTeXe The \LaTeX<sub>2ε</sub> logo as proposed by A-W designers.

```
13 \DeclareRobustCommand{\LaTeXe}{\mbox{\m@th
14 \if b\expandafter\@car\f@series\@nil\boldmath\fi
15 \LaTeX\kern.15em\textstyle\varepsilon\}}%
16 </2ekernel>
```

(End of definition for \LaTeXe.)

# File r

## ltfiles.dtx

### 1 File Handling

The following user commands are defined in this part:

|                    |                                                                                                                                                                                             |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \document          | (ie \begin{document})                                                                                                                                                                       |
|                    | Reads in the .AUX files and \catcode's @ to 12.                                                                                                                                             |
| \nofiles           | Suppresses all file output by setting \@filesw false.                                                                                                                                       |
| \includeonly       | \{(NAME1, ... ,NAMEn)\}                                                                                                                                                                     |
|                    | Causes only parts NAME1, ... ,NAMEn to be read by their \include commands. Works by setting partsw true and setting \@partlist to NAME1, ... ,NAMEn.                                        |
| \include           | \{(NAME)\}                                                                                                                                                                                  |
|                    | Does an \input NAME unless \@partsw is true and NAME is not in \@partlist. If \@filesw is true, then it directs .AUX output to NAME.AUX, including a checkpoint at the end.                 |
| \input             | \{(NAME)\}                                                                                                                                                                                  |
|                    | The same as TeX's \input, except it allows optional braces around the file name. In LATEX 2 <sub>E</sub> , it also avoids the primitive 'missing file' error, if the file can not be found. |
| \IfFileExists      | \{(NAME)\}\{(then)\}\{(else)\}                                                                                                                                                              |
|                    | If the file exists on the system, execute then otherwise execute else.                                                                                                                      |
| \InputIfFileExists | \{(NAME)\}\{(then)\}\{(else)\}                                                                                                                                                              |
|                    | If the file exists on the system, execute then and input NAME otherwise execute else.                                                                                                       |

*Historical LATEX 2.09 comments (not necessarily accurate any more):*

1 <\*2ekernel>  
2 \message{files,}

#### VARIABLES, SWITCHES AND INTERNAL COMMANDS:

|             |                                                                                                 |
|-------------|-------------------------------------------------------------------------------------------------|
| @mainaux    | : Output file number for main .AUX file.                                                        |
| @partaux    | : Output file number for current part's .AUX file.                                              |
| @auxout     | : Either @mainout or @partout, depending on which .AUX file output goes to.                     |
| @input{foo} | : If file foo exists, then \input's it, otherwise types a warning message.                      |
| @filesw     | : Switch – set false if no .AUX, .TOC, .IDX etc files are to be written                         |
| @partsw     | : Set true by a \includeonly command.                                                           |
| @partlist   | : Set to the argument of the \includeonly command.                                              |
| \cp@FOO     | : The checkpoint for \include'd file FOO.TEX, written by \@writeckpt at the end of file FOO.AUX |

\includeonly{FILELIST} ==  
BEGIN

```

\@partsw := T
\@partlist := FILELIST
END

\include{FILE} ==
BEGIN
 \clearpage
 if \@files w = T
 then \immediate\write\@mainaux{\string\@input{FILE.AUX}}
 fi
 if \@partsw = T
 then \@tempswa := F
 \reserved@b == FILE
 for \reserved@a := \@partlist
 do if eval(\reserved@a) = eval(\reserved@b)
 then \@tempswa := T fi
 od
 fi

 if \@tempswa = T
 then \@auxout := \@partaux
 if \@files w = T
 then \immediate\openout\@partaux{FILE.AUX}
 \immediate\write\@partaux{\relax}
 fi
 \@input{FILE.TEX}
 \clearpage
 \@writeckpt{FILE}
 if @files w then \closeout\@partaux fi
 \@auxout := \@mainaux
 else \cp@FILE
 fi
END

\@writeckpt{FILE} ==
BEGIN
 if \@files w = T
 \immediate\write on file \@partaux:
 \@setckpt{FILE}{% }
 for \reserved@a := \cl@ckpt
 do \immediate\write on file \@partaux:
 \global\string\setcounter
 {eval(\reserved@a)}{eval(\c@eval(\reserved@a))}
 od
 \immediate\write on file \@partaux: %
 fi
END

\@setckpt{FILE}{LIST} ==
BEGIN

```

```
G \cp@FILE := LIST
END
```

```
INITIALIZATION
\@tempswa := T
```

*End of historical L<sup>A</sup>T<sub>E</sub>X 2.09 comments.*

```
\@mainaux
\@partaux
 3 \newwrite\@mainaux
 4 \newwrite\@partaux
```

*(End of definition for \@mainaux and \@partaux.)*

```
\if@filesw
\if@partsw
 5 \newif\if@filesw \@fileswtrue
 6 \newif\if@partsw \@partswfalse
```

*(End of 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 of 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 \expl@sys@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 \iffirstcolumntrue
29 \fi
30 \hsize\columnwidth \linewidth\hsize
31 \begingroup\@floatplacement\@dblfloatplacement
32 \makeatletter\let\@writefile\gobbletwo
33 \global \let \multiplelabels \relax
34 \cinput{\jobname.aux}%
35 \endgroup
36 \if@files
37 \immediate\openout\mainaux\jobname.aux
38 \immediate\write\mainaux{\relax}%
39 \fi

```

Dateline 1991/03/26: FMi added `\process@table` to support NFSS; This will also work with old lfonts if no other style defines `\process@table`. The following line forces the initialization of the math fonts.

```

40 \process@table
41 \let\glb@currsize\empty % Force math initialization.

42 \normalsize
43 \everypar{}%

```

So that punctuation in headings is not disturbed by verbatim or other local changes to the space factor codes, save the document default here. This will be locally reset by the output routine. For special cases a class may want to define `\normalsfcodes` directly, in case that definition will be used. (This is an old bug, problem existed in L<sup>A</sup>T<sub>E</sub>X2.0x and plain T<sub>E</sub>X.)

```

44 \ifx\normalsfcodes\empty
45 \ifnum\sfcodes`_=\@m
46 \let\normalsfcodes\frenchspacing
47 \else
48 \let\normalsfcodes\nonfrenchspacing
49 \fi
50 \fi

```

For similar reasons also save the default language, this will be reset locally in the output routine. In particular it allows hyphenation in the page head even if the page break happens in verbatim. If this has already been set by a package, set to the value of `\language` at this point.

```

51 \ifx\document@default@language\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

```

Provide a global definition for `\do` as well, so that it is already defined in the preamble and not late as `\begin{document}` overwriting some definition given by the unsuspecting user in the preamble.

```

72 \let\do\noexpand

```

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`.

```

73 \NewHook{begindocument}
74 \NewHook{begindocument/before}
75 \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.

```

76 \edef \@kernel@after@begindocument@before {%
77 \let\expandafter\noexpand\csname
78 __hook env/document/begin\endcsname
79 \noexpand\empty}
80 %\let \@kernel@before@begindocument \empty
81 %\let \@kernel@after@begindocument \empty
82 (./2ekernel | latexrelease)
83 (latexrelease)\EndIncludeInRelease
84 (latexrelease)\IncludeInRelease{2017/04/15}%
85 (latexrelease) {\document}{Save language for hyphenation}%
86 (latexrelease)
87 (latexrelease)\def\document{\endgroup
88 (latexrelease) \ifx\@unusedoptionlist\empty\else
89 (latexrelease) \@latex@warning@no@line{Unused global option(s):`^' }%
90 (latexrelease) \@spaces[\@unusedoptionlist]}%
91 (latexrelease) \fi
92 (latexrelease) \colht\textheight
93 (latexrelease) \colroom\textheight \vsize\textheight
94 (latexrelease) \columnwidth\textwidth
95 (latexrelease) \clubpenalty\clubpenalty
96 (latexrelease) \if@twocolumn
97 (latexrelease) \advance\columnwidth -\columnsep
98 (latexrelease) \divide\columnwidth\tw@ \hsize\columnwidth \iffirstcolumntrue
99 (latexrelease) \fi
100 (latexrelease) \hsize\columnwidth \linewidth\hsize
101 (latexrelease) \begingroup\flo@placement\dblfloatplacement
102 (latexrelease) \makeatletter\let\@writetwo\gobbletwo
103 (latexrelease) \global\let\@multiplelabels\relax
104 (latexrelease) \input{\jobname.aux}%
105 (latexrelease) \endgroup
106 (latexrelease) \if@filesw
107 (latexrelease) \immediate\openout\mainaux\jobname.aux
108 (latexrelease) \immediate\write\mainaux{\relax}%
109 (latexrelease) \fi
110 (latexrelease) \process@table
111 (latexrelease) \let\glb@currsize\empty % Force math initialization.
112 (latexrelease) \normalsize
113 (latexrelease) \everypar{}%
114 (latexrelease) \ifx\normalsfcodes\empty
115 (latexrelease) \ifnum\sfcodes`.=\@m
116 (latexrelease) \let\normalsfcodes\frenchspacing
117 (latexrelease) \else
118 (latexrelease) \let\normalsfcodes\nonfrenchspacing
119 (latexrelease) \fi
120 (latexrelease) \fi
121 (latexrelease) \ifx\document@default@language\m@ne
122 (latexrelease) \chardef\document@default@language\language
123 (latexrelease) \fi

```

```

124 〈latexrelease〉 \c@noskipsecfalse
125 〈latexrelease〉 \let \orefundefined \relax
126 〈latexrelease〉 \let\AtBeginDocument\@firstofone
127 〈latexrelease〉 \begindocumenthook
128 〈latexrelease〉 \ifdim\topskip<1sp\global\topskip 1sp\relax\fi
129 〈latexrelease〉 \global\c@maxdepth\maxdepth
130 〈latexrelease〉 \global\let\@begindocumenthook\@undefined
131 〈latexrelease〉 \ifx\@listfiles\@undefined
132 〈latexrelease〉 \global\let\@filelist\relax
133 〈latexrelease〉 \global\let\@addtofilelist\@gobble
134 〈latexrelease〉 \fi
135 〈latexrelease〉 \gdef\do##1{\global\let ##1\@notprerr}%
136 〈latexrelease〉 \preamblecmds
137 〈latexrelease〉 \global\let \c@nodocument \relax
138 〈latexrelease〉 \global\let\do\noexpand
139 〈latexrelease〉 \ignorespaces}
140 〈latexrelease〉 \EndIncludeInRelease
141 〈latexrelease〉
142 〈latexrelease〉 \IncludeInRelease{0000/00/00}%
143 〈latexrelease〉 {\document}{Save language for hyphenation}
144 〈latexrelease〉 \def\document{\endgroup
145 〈latexrelease〉 \ifx\@unusedoptionlist\empty\else
146 〈latexrelease〉 \c@latex@warning@no@line{Unused global option(s):`^J`%
147 〈latexrelease〉 \c@spaces[\@unusedoptionlist]}%
148 〈latexrelease〉 \fi
149 〈latexrelease〉 \c@colht\textheight
150 〈latexrelease〉 \c@colroom\textheight \vsize\textheight
151 〈latexrelease〉 \columnwidth\textwidth
152 〈latexrelease〉 \clubpenalty\clubpenalty
153 〈latexrelease〉 \if@twocolumn
154 〈latexrelease〉 \advance\columnwidth -\columnsep
155 〈latexrelease〉 \divide\columnwidth\tw@ \hsize\columnwidth
156 〈latexrelease〉 \c@firstcolumntrue
157 〈latexrelease〉 \fi
158 〈latexrelease〉 \hsize\columnwidth \linewidth\hsize
159 〈latexrelease〉 \begingroup\@floatplacement\@dblfloatplacement
160 〈latexrelease〉 \makeatletter\let\c@writefile\@gobbletwo
161 〈latexrelease〉 \global\let\c@multipletables\relax
162 〈latexrelease〉 \c@input{\jobname.aux}%
163 〈latexrelease〉 \endgroup
164 〈latexrelease〉 \if@filesw
165 〈latexrelease〉 \immediate\openout\c@mdeaux\jobname.aux
166 〈latexrelease〉 \immediate\write\c@mdeaux{\relax}%
167 〈latexrelease〉 \fi
168 〈latexrelease〉 \process@table
169 〈latexrelease〉 \let\glb@currsize\empty
170 〈latexrelease〉 \normalsize
171 〈latexrelease〉 \everypar{}%
172 〈latexrelease〉 \ifx\normalsfcodes\empty
173 〈latexrelease〉 \ifnum\sfcodes`.=\@m
174 〈latexrelease〉 \let\normalsfcodes\frenchspacing
175 〈latexrelease〉 \else
176 〈latexrelease〉 \let\normalsfcodes\nonfrenchspacing
177 〈latexrelease〉 \fi

```

```

178 \if{latexrelease} \fi
179 \if{latexrelease} \@noskipsecfalse
180 \if{latexrelease} \let \erefundefined \relax
181 \if{latexrelease} \let\AtBeginDocument\@firstofone
182 \if{latexrelease} \begindocumenthook
183 \if{latexrelease} \ifdim\topskip<1sp\global\topskip 1sp\relax\fi
184 \if{latexrelease} \global\@maxdepth\maxdepth
185 \if{latexrelease} \global\let\@begindocumenthook\@undefined
186 \if{latexrelease} \ifx\@listfiles\@undefined
187 \if{latexrelease} \global\let\@filelist\relax
188 \if{latexrelease} \global\let\@addtofilelist\@gobble
189 \if{latexrelease} \fi
190 \if{latexrelease} \gdef\do##1{\global\let ##1\@notprerr}%
191 \if{latexrelease} \preamblecmds
192 \if{latexrelease} \global\let\@nodocument \relax
193 \if{latexrelease} \global\let\do\noexpand
194 \if{latexrelease} \ignorespaces}
195 \if{latexrelease} \EndIncludeInRelease
196 {*2ekernel}

197 \onlypreamble\document

```

(End of 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.

```
198 \let\normalsfcodes\@empty
```

(End of definition for *\normalsfcodes*.)

**\nofiles** Set *\@fileswfalse* which suppresses the places where L<sup>A</sup>T<sub>E</sub>X 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.

```

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

```

(End of 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*.

```

207 \long\def \protected@write#1#2#3{%
208 \begingroup
209 \let\thepage\relax
210 #2%
211 \let\protect\@unexpandable@protect
212 \edef\reserved@a{\write#1{#3}}%

```

```

213 \reserved@a
214 \endgroup
215 \if@nobreak\ifvmode\nobreak\fi\fi
216 }

```

(End of definition for `\protected@write`.)

```
217 \let\@auxout=\@mainaux
```

`\include` In the definition of `\include`, `\def\reserved@b` changed to `\edef\reserved@b` to be  
`\includeonly` consistent with the `\edef` in `\includeonly`. (Suggested by Rainer Schöpf & Frank  
Mittelbach. Change made 20 Jul 88.)

Changed definition of `\include` to allow space at end of file name — otherwise,  
typing `\include{foo }` would cause L<sup>A</sup>T<sub>E</sub>X to overwrite `foo.tex`. Change made 24 May  
89, suggested by Rainer Schöpf and Frank Mittelbach

Made `\include` check for being used inside an `\include`'d file, as this will not work  
and cause surprising results.

```

218 </2ekernel>
219 <*2ekernel | latexrelease>
220 <| latexrelease>\IncludeInRelease{2020/10/01}%
221 <| latexrelease> {\includeonly}{Spaces in file names}%
222 \def\include#1{\relax
223 \ifnum\@auxout=\@partaux
224 \@latex@error{\string\include\space cannot be nested}\@eha
225 \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.

```

226 \set@curr@file{#1}%
227 \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.

```

228 \expandafter\@include\expandafter{\@curr@file} % deliberate space
229 \fi}

```

Here in `\includeonly` we also need to strip `.tex` after normalization:

```

230 \def\includeonly#1{%
231 \@partstrue

```

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.

```

232 \let\@partlist\@empty
233 \@for\reserved@a:=#1 \do
234 {%
235 \expandafter\set@curr@file\expandafter{\reserved@a}%
236 \ifx\@partlist\@empty
237 \edef\@partlist{\@strip@tex@ext\@curr@file}%
238 \else
239 \edef\@partlist{\@partlist,\@strip@tex@ext\@curr@file}%
240 \fi
241 }%
242 }
243 \onlypreamble\includeonly

```

(End of definition for \include and \includeonly.)

\@strip@tex@ext These macros take a (\detokenized file name and remove any .tex extension). Extra care is taken to not remove the string .tex from the middle of a file name: it is only removed if it's the very last thing in the file name.

```
244 \def\reserved@a#1{%
245 \@strip@tex@ext##1{%
246 \expandafter\@strip@tex@ext@aux
247 ##1\@nil\@nil
248 #1\@nil\relax\@nnil}
249 \def\@strip@tex@aux##1#1\@nil##2\@nnil{%
250 \ifx\relax##2\@empty
251 \expandafter\@cdr\expandafter\@empty\@cdr{}##1%
252 \else##1\fi}%
253 \expandafter\reserved@a
254 \expandafter{\detokenize{.tex}}
255
```

(End of definition for \@strip@tex@ext and \@strip@tex@ext@aux.)

```
256 \end{tex} \EndIncludeInRelease
257 \end{tex} \IncludeInRelease{2019/10/01}%
258 \end{tex} {\includeonly}{Spaces in file names}%
259 \end{tex}
260 \def\includeonly#1{%
261 \partswtrue
262 \set@curr@file{\zap@space#1 \empty}%
263 \let\@partlist\curr@file
264 }
265 \end{tex}
266 \def\include#1{\relax
267 \ifnum\auxout=\partaux
268 \@latex@error{\string\include\space cannot be nested}\@eha
269 \else
270 \set@curr@file{#1 }%
271 \expandafter\@include\curr@file
272 \fi}
273 \end{tex}
274 \let\@strip@tex@ext\undefined
275 \let\@strip@tex@ext@aux\undefined
276 \end{tex}
277 \end{tex} \EndIncludeInRelease
278 \IncludeInRelease{0000/00/00}%
279 \end{tex} {\includeonly}{Spaces in file names}%
280 \def\includeonly#1{%
281 \partswtrue
282 \edef\@partlist{\zap@space#1 \empty}%
283 }
284 \def\include#1{\relax
285 \ifnum\auxout=\partaux
286 \@latex@error{\string\include\space cannot be nested}\@eha
287 \else \include#1 \fi}
288 \end{tex}
289 \end{tex} \EndIncludeInRelease
290
```

```

\@include
291 </2ekernel>
292 {*2ekernel | latexrelease}
293 {latexrelease}\IncludeInRelease{2022/06/01}%
294 {latexrelease} {\@include}{Spaces in file names and hooks}%

295 \def\@include#1 {%
296 \ifx\@nodocument\relax
297 \clearpage
298 \if@filesw
299 \immediate\write\@mainaux{\string\@input{#1.aux}}%
300 \fi
301 \tempswattrue
302 \if@partsw
303 \tempswafalse
304 \edef\reserved@b{#1}%
305 \for\reserved@a:=\partlist\do
306 {\ifx\reserved@a\reserved@b\tempswattrue\fi}%
307 \fi
308 \if@tempswa
309 \let\auxout\partaux
310 \if@filesw
311 \immediate\openout\partaux "#1.aux"
312 \immediate\write\partaux{\relax}%
313 \fi

```

Now before going to the hooks we need to set `\CurrentFile`:

```

314 %-----%
315 \filehook@set@CurrentFile

```

Execute the `before` hooks just after we switched the `.aux` file ...

```

316 \UseHook{include/before}%
317 \UseOneTimeHook{include/#1/before}%
318 %-----%
319 \input{#1.tex}%
320 %-----%
... then end hooks ...
321 \UseOneTimeHook{include/#1/end}%
322 \UseHook{include/end}%
323 %-----%
324 \clearpage
325 %-----%

```

... 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`).

```

326 \UseOneTimeHook{include/#1/after}%
327 \UseHook{include/after}%
328 \clearpage
329 %-----%
330 \writeckpt{#1}%
331 \if@filesw
332 \immediate\closeout\partaux

```

```

333 \fi
334 \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.

```

335 \deadcycles\z@
336 \curnameuse{cp@#1}\%
```

We also execute a hook in this case, first a general used for every include file that is exclude and then a specific one that contains the name of the include file.

```

337 %-----%
338 \UseHook{include/excluded}%
339 \UseOneTimeHook{include/#1/excluded}%
340 %-----%
341 \fi
342 \let\@auxout\@mainaux
343 \else
344 \@latex@warning{%
345 \noexpand\include should only be used after \string\begin{document}}%
346 \cinput@{#1}%
347 \fi}
```

Now declare the non-generic `include` hooks used above:

```

348 \NewHook{include/before}
349 \NewReversedHook{include/end}
350 \NewReversedHook{include/after}
351 \NewHook{include/excluded}
352 <latexrelease> \EndIncludeInRelease
353 (/2ekernel | latexrelease)

354 <latexrelease> \IncludeInRelease{2020/10/01}%
355 <latexrelease> {:@include}{Spaces in file names and hooks}%
356 <latexrelease> \EndIncludeInRelease
357 <latexrelease> \def\@include#1 {%
358 <latexrelease> \ifx\@nodocument\relax
359 <latexrelease> \clearpage
360 <latexrelease> \if@filesw
361 <latexrelease> \immediate\write\@mainaux{\string\@input{#1.aux}}%
362 <latexrelease> \fi
363 <latexrelease> \tempswattrue
364 <latexrelease> \if@partsw
365 <latexrelease> \tempswafalse
366 <latexrelease> \edef\reserved@b{#1}%
367 <latexrelease> \for\reserved@a:=\partlist\do
368 <latexrelease> {\ifx\reserved@a\reserved@b\tempswattrue\fi}%
369 <latexrelease> \fi
370 <latexrelease> \if@tempswa
371 <latexrelease> \let\@auxout\partaux
372 <latexrelease> \if@filesw
373 <latexrelease> \immediate\openout\partaux "#1.aux"
374 <latexrelease> \immediate\write\partaux{\relax}%
375 <latexrelease> \fi
376 <latexrelease> \filehook@set@CurrentFile
377 <latexrelease> \UseHook{include/before}%
378 <latexrelease> \UseOneTimeHook{include/#1/before}%

```

```

379 <{latexrelease}> \@input@{#1.tex}%
380 <{latexrelease}> \UseOneTimeHook{include/#1/end}%
381 <{latexrelease}> \UseHook{include/end}%
382 <{latexrelease}> \clearpage
383 <{latexrelease}> \UseOneTimeHook{include/#1/after}%
384 <{latexrelease}> \UseHook{include/after}%
385 <{latexrelease}> \clearpage
386 <{latexrelease}> \@writeckpt{#1}%
387 <{latexrelease}> \if@filesw
388 <{latexrelease}> \immediate\closeout\@partaux
389 <{latexrelease}> \fi
390 <{latexrelease}> \else
391 <{latexrelease}> \deadcycles\z@
392 <{latexrelease}> \@nameuse{cp@#1}%
393 <{latexrelease}> \fi
394 <{latexrelease}> \let\@auxout\@mainaux
395 <{latexrelease}\else
396 <{latexrelease}\@latex@warning{%
397 <{latexrelease}> \noexpand\include should only be used after \string\begin{document}}%
398 <{latexrelease}\@input@{#1}%
399 <{latexrelease}\fi}
400 <{latexrelease}\NewHook{include/before}%
401 <{latexrelease}\NewReversedHook{include/end}%
402 <{latexrelease}\NewReversedHook{include/after}%
403 <{latexrelease}\IncludeInRelease{0000/00/00}%
404 <{latexrelease}> {\@include}{Spaces in file names and hooks}%
405 <{latexrelease}\def\@include#1 {%
406 <{latexrelease}> \clearpage
407 <{latexrelease}> \if@filesw
408 <{latexrelease}> \immediate\write\@mainaux{\string\@input{#1.aux}}%
409 <{latexrelease}> \fi
410 <{latexrelease}> \tempswattrue
411 <{latexrelease}> \if@partsw
412 <{latexrelease}> \tempswafalse
413 <{latexrelease}> \edef\reserved@b{#1}%
414 <{latexrelease}> \for\reserved@a:=\partlist\do
415 <{latexrelease}> \ifx\reserved@a\reserved@b\tempswattrue\fi}%
416 <{latexrelease}> \fi
417 <{latexrelease}> \if@tempswa
418 <{latexrelease}> \let\@auxout\@partaux
419 <{latexrelease}> \if@filesw
420 <{latexrelease}> \immediate\openout\@partaux #1.aux
421 <{latexrelease}> \immediate\write\@partaux{\relax}%
422 <{latexrelease}> \fi
423 <{latexrelease}> \@input@{#1.tex}%
424 <{latexrelease}> \clearpage
425 <{latexrelease}> \@writeckpt{#1}%
426 <{latexrelease}> \if@filesw
427 <{latexrelease}> \immediate\closeout\@partaux
428 <{latexrelease}> \fi
429 <{latexrelease}> \else
430 <{latexrelease}> \deadcycles\z@
431 <{latexrelease}> \@nameuse{cp@#1}%
432 <{latexrelease}> \fi

```

```

433 〈latexrelease〉 \let\@auxout\@mainaux}
434 〈latexrelease〉
435 〈latexrelease〉\EndIncludeInRelease
436 〈*2ekernel〉

```

(End of definition for \@include.)

\@writeckpt

```

437 \def\@writeckpt#1{%
438 \if@filesw
439 \immediate\write\@partaux{\string\@setckpt{\#1}\@charlb}%
440 {\let\@elt\@wckptelt \cl@0@ckpt}%
441 \immediate\write\@partaux{\@charrb}%
442 \fi}

```

(End of definition for \@writeckpt.)

\@wckptelt

```

443 \def\@wckptelt#1{%
444 \immediate\write\@partaux{%
445 \string\setcounter{\#1}{\the\@nameuse{c@\#1}}}}

```

(End of definition for \@wckptelt.)

\@setckpt RmS 93/08/31: introduced \@setckpt

```
446 \def\@setckpt#1{\global\@namedef{cp@\#1}}
```

(End of definition for \@setckpt.)

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

```

447 {\catcode`[=1 \catcode`=2
448 \catcode`{=11 \catcode`}=11
449 \gdef\@charlb[{]
450 \gdef\@charrb[]}
451 }% }brace matching

```

(End of 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.

```

452 \def\set@curr@file#1{%
453 \begingroup
454 \escapechar\m@ne
455 \xdef\@curr@file{%
456 \expandafter\expandafter\expandafter\unquote@name
457 \expandafter\expandafter\expandafter{\%
458 \expandafter\string
459 \csname@\firstofone#1\empty\endcsname}%
460 \endgroup
461 }

```

(End of definition for `\@curr@file` and `\set@curr@file`.)

|                            |                |
|----------------------------|----------------|
| <code>\quote@name</code>   | Quoting spaces |
| <code>\quote@@name</code>  |                |
| <code>\unquote@name</code> |                |
| a b c                      | -> "a b c"     |
| "a b c"                    | -> "a b c"     |
| a" "b" "c                  | -> "a b c"     |
|                            | -> ""          |

```

462 </2ekernel>
463 <*2ekernel | latexrelease>
464 <latexrelease>\IncludeInRelease{2019/10/01}%
465 <latexrelease> {\quote@name}{Quote file names}%
466 \def\quote@name#1{"\quote@@name#1\@gobble"}%
467 \def\quote@@name#1"#1\quote@@name}

```

and removing quotes ...

```
468 \def\unquote@name#1{\quote@@name#1\@gobble"}
```

(End of definition for `\quote@name`, `\quote@@name`, and `\unquote@name`.)

|                            |  |
|----------------------------|--|
| <code>\IfFileExists</code> |  |
|----------------------------|--|

```

469 \DeclareRobustCommand\IfFileExists[1]{%
470 \set@curr@file{#1}%
471 \expandafter\IfFileExists@\expandafter{\@curr@file}}

```

(End of definition for `\IfFileExists`.)

```

472 </2ekernel | latexrelease>
473 <latexrelease>\EndIncludeInRelease
474 <latexrelease>\IncludeInRelease{0000/00/00}%
475 <latexrelease> {\quote@name}{Quote file names}%
476 <latexrelease>
477 <latexrelease>\let\quote@name\@undefined
478 <latexrelease>\let\quote@@name\@undefined
479 <latexrelease>\let\unquote@name\@undefined
480 <latexrelease>
481 <latexrelease>\long\def \IfFileExists#1#2#3{%
482 <latexrelease> \openin\@inputcheck#1 %
483 <latexrelease> \ifeof\@inputcheck
484 <latexrelease> \ifx\input@path\@undefined
485 <latexrelease> \def\reserved@a{#3}%
486 <latexrelease> \else

```

```

487 〈\latexrelease〉 \def\reserved@a{\@iffileonpath{#1}{#2}{#3}}%
488 〈\latexrelease〉 \fi
489 〈\latexrelease〉 \else
490 〈\latexrelease〉 \closein\@inputcheck
491 〈\latexrelease〉 \edef\@filef@und{#1 }%
492 〈\latexrelease〉 \def\reserved@a{#2}%
493 〈\latexrelease〉 \fi
494 〈\latexrelease〉 \reserved@a}
495 〈\latexrelease〉
496 〈\latexrelease〉\EndIncludeInRelease
497 (*2ekernel〉

```

\IfFileExists@ Argument #1 is \curr@file so catcode 12 string with no quotes.  
\IfFileExists@@ 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.

Earlier versions used \openin here, but this led to two code paths, one in expl3 and one here. To avoid that, and as the expl3 approach works by expansion, we use that here. As we need the file name to include the path, the actual expl3 function used is not the file existence test!

```

498 〈/2ekernel〉
499 〈*2ekernel | \latexrelease〉
500 〈\latexrelease〉\IncludeInRelease{2023/05/01}%
501 〈\latexrelease〉 {\IfFileExists@}{\IfFileExists}%
502 \long\def \IfFileExists@#1#2#3{%
503 \edef\@filef@und{\IfFileExists@#1}%

```

The expl3 function regards an empty argument as nothing at all, but the L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\varepsilon$</sub>  convention is that this is equal to the special .tex file.

```

504 \ifx\@filef@und\@empty
505 \if\relax\detokenize{#1}\relax
506 \let\reserved@a\@firstoftwo
507 \def\@filef@und{"tex"}%
508 \else
509 \let\reserved@a\@secondoftwo
510 \fi
511 \else
512 \let\reserved@a\@firstoftwo
513 \edef\@filef@und"\@filef@und" %
514 \fi

```

This is just there so that any # inside #2 or #3 needs doubling (as that was the case in the past).

```

515 \expandafter\def\expandafter\reserved@a
516 \expandafter{\reserved@a{#2}{#3}}%
517 \reserved@a

```

Pipes are not really files, but at the document level they are supported. To quickly trim of any leading spaces, we use a blank test and \use:n rather than \tl\_trim\_spaces:n for speed as we don't care about the end of the input.

```

518 \ExplSyntaxOn
519 \cs_new:Npn \IfFileExists@@ #1

```

```

520 {
521 \tl_if_blank:nF {#1}
522 {
523 \tl_if_head_eq_charcode:oNTF { \use:n #1 } |
524 {#1}
525 { \file_full_name:n {#1} }
526 }
527 }
528 \cs_generate_variant:Nn \tl_if_head_eq_charcode:nNTF { o }
529 \ExplSyntaxOff
530 ⟨/2ekernel | latexrelease⟩
531 ⟨latexrelease⟩\EndIncludeInRelease
532 ⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
533 ⟨latexrelease⟩
534 ⟨latexrelease⟩
535 ⟨latexrelease⟩\long\def \IfFileExists@#1#2#3{%
536 ⟨latexrelease⟩ \openin\@inputcheck"#1" %
537 ⟨latexrelease⟩ \ifeof\@inputcheck
538 ⟨latexrelease⟩ \ifx\input@path\@undefined
539 ⟨latexrelease⟩ \let\reserved@a\@secondoftwo
540 ⟨latexrelease⟩ \else
541 ⟨latexrelease⟩ \def\reserved@a{\@iffilenonpath{#1}}%
542 ⟨latexrelease⟩ \fi
543 ⟨latexrelease⟩ \else
544 ⟨latexrelease⟩ \closein\@inputcheck
545 ⟨latexrelease⟩ \edef\@filef@und{"#1"}%
546 ⟨latexrelease⟩ \let\reserved@a\@firstoftwo
547 ⟨latexrelease⟩ \fi
548 ⟨latexrelease⟩ \expandafter\def\expandafter\reserved@a
549 ⟨latexrelease⟩ \expandafter{\reserved@a{#2}{#3}}%
550 ⟨latexrelease⟩\reserved@a
551 ⟨latexrelease⟩\let\IfFileExists@@\@undefined
552 ⟨latexrelease⟩\EndIncludeInRelease
553 ⟨latexrelease⟩
554 ⟨latexrelease⟩\IncludeInRelease{2019/10/01}%
555 ⟨latexrelease⟩
556 ⟨latexrelease⟩
557 ⟨latexrelease⟩\long\def \IfFileExists@#1#2#3{%
558 ⟨latexrelease⟩ \openin\@inputcheck"#1" %
559 ⟨latexrelease⟩ \ifeof\@inputcheck
560 ⟨latexrelease⟩ \ifx\input@path\@undefined
561 ⟨latexrelease⟩ \def\reserved@a{#3}%
562 ⟨latexrelease⟩ \else
563 ⟨latexrelease⟩ \def\reserved@a{\@iffilenonpath{#1}{#2}{#3}}%
564 ⟨latexrelease⟩ \fi
565 ⟨latexrelease⟩ \else
566 ⟨latexrelease⟩ \closein\@inputcheck
567 ⟨latexrelease⟩ \edef\@filef@und{"#1"}%
568 ⟨latexrelease⟩ \def\reserved@a{#2}%
569 ⟨latexrelease⟩ \fi
570 ⟨latexrelease⟩ \reserved@a
571 ⟨latexrelease⟩\EndIncludeInRelease
572 ⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
573 ⟨latexrelease⟩

```

```

574 〈\latexrelease〉
575 〈\latexrelease〉\let\IfFileExists@{\@undefined
576 〈\latexrelease〉
577 〈\latexrelease〉
578 〈\latexrelease〉\EndIncludeInRelease
579 {<2ekernel〉

(End of definition for \IfFileExists@ and \IfFileExists@@.)
```

**\@iffileonpath** If the file is not found by \openin, and \input@path is defined, look in all the directories specified in \input@path.

```

580 {</2ekernel>
581 {<2ekernel | \latexrelease>
582 〈\latexrelease〉\IncludeInRelease{2019/10/01}%
583 〈\latexrelease〉{\@iffileonpath}{Quote file names}
584 \long\def\@iffileonpath#1{%
585 \let\reserved@a\@secondoftwo
586 \expandafter\@tfor\expandafter\reserved@b\expandafter
587 :\expandafter=\input@path\do{%
588 \openin\@inputcheck\expandafter\quote@name\expandafter{\reserved@b#1} %
589 \ifeof\@inputcheck\else
590 \edef\@filef@und{\expandafter\quote@name\expandafter{\reserved@b#1} }%
591 \let\reserved@a\@firstoftwo%
592 \closein\@inputcheck
593 \@break@tfor
594 \fi}%
595 \reserved@a}
```

(End of definition for \@iffileonpath.)

```

596 {</2ekernel | \latexrelease>
597 〈\latexrelease〉\EndIncludeInRelease
598 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
599 〈\latexrelease〉{\@quote@name}{Quote file names}
600 〈\latexrelease〉
601 〈\latexrelease〉\long\def\@iffileonpath#1{%
602 \let\reserved@a\@secondoftwo
603 \expandafter\@tfor\expandafter\reserved@b\expandafter
604 :\expandafter=\input@path\do{%
605 \openin\@inputcheck\reserved@b#1 %
606 \ifeof\@inputcheck\else
607 \edef\@filef@und{\reserved@b#1 }%
608 \let\reserved@a\@firstoftwo%
609 \closein\@inputcheck
610 \@break@tfor
611 \fi}%
612 \reserved@a}
613 〈\latexrelease〉
614 〈\latexrelease〉\EndIncludeInRelease
615 {<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**.

```

616 \DeclareRobustCommand \InputIfFileExists[2]{%
617 \IfFileExists{#1}{%
618 {%
619 \expandafter\@swaptwoargs\expandafter
620 {\@filef@und}{#2\@addtofilelist{#1}\@@input}}}

```

*(End of definition for \InputIfFileExists.)*

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

```

621 </2ekernel>
622 <*2ekernel | latexrelease>
623 <latexrelease>\IncludeInRelease{2019/10/01}%
624 <latexrelease> {\@swaptwoargs}{Don't lose the file name}%
625 \long\def\@swaptwoargs#1#2{#2#1}

626 </2ekernel | latexrelease>
627 <latexrelease>\EndIncludeInRelease
628 <latexrelease>\IncludeInRelease{0000/00/00}%
629 <latexrelease> {\@swaptwoargs}{Don't lose the file name}%
630 <latexrelease>\let\@swaptwoargs\@undefined
631 <latexrelease>\EndIncludeInRelease
632 <*2ekernel>

```

*(End of definition for \@swaptwoargs.)*

**\input** Input a file: if the argument is given in braces use safe input macros, otherwise use TeX's primitive \input command (which is called \@@input in L<sup>A</sup>T<sub>E</sub>X).

```

633 \def\input{\@ifnextchar\bgroup\@iinput\@@input}

```

*(End of definition for \input.)*

**\@iinput** Define \@iinput (i.e., \input) in terms of \InputIfFileExists.

Changes to \@iinput: adapt to the changes to \@missingfileerror.

```

634 </2ekernel>
635 <*2ekernel | latexrelease>
636 <latexrelease>\IncludeInRelease{2020/10/01}%
637 <latexrelease> {\@iinput}{Change in file error handling}%
638 \def\@iinput#1{%
639 \InputIfFileExists{#1}{}%
640 {\filename@parse\@curr@file
641 \edef\reserved@a{\noexpand\@missingfileerror
642 {\filename@area\filename@base}%
643 {\ifx\filename@ext\relax tex\else\filename@ext\fi}}%

```

This line now just sets \@missingfile@{*part*}:

```

644 \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.

```

645 \edef\reserved@a{\noexpand\@iinput{%
646 \@missingfile@area\@missingfile@base.\@missingfile@ext} }%
647 \reserved@a}%
648 </2ekernel | latexrelease>
649 <latexrelease>\EndIncludeInRelease

```

```

650 〈latexrelease〉\IncludeInRelease{2019/10/01}%
651 〈latexrelease〉 {\@iinput}{Quote file names}%
652 〈latexrelease〉
653 〈latexrelease〉\def\@iinput#1{%
654 〈latexrelease〉 \InputIfFileExists{#1}{}%
655 〈latexrelease〉 {\filename@parse@\curr@file
656 〈latexrelease〉 \edef\reserved@a{\noexpand\@missingfileerror
657 〈latexrelease〉 {\filename@area\filename@base}%
658 〈latexrelease〉 {\ifx\filename@ext\relax tex\else\filename@ext\fi} }%
659 〈latexrelease〉 \reserved@a}%
660 〈latexrelease〉\EndIncludeInRelease
661 〈latexrelease〉\IncludeInRelease{0000/00/00}%
662 〈latexrelease〉 {\@iinput}{Quote file names}%
663 〈latexrelease〉\def\@iinput#1{%
664 〈latexrelease〉 \InputIfFileExists{#1}{}%
665 〈latexrelease〉 {\filename@parse{#1}%
666 〈latexrelease〉 \edef\reserved@a{\noexpand\@missingfileerror
667 〈latexrelease〉 {\filename@area\filename@base}%
668 〈latexrelease〉 {\ifx\filename@ext\relax tex\else\filename@ext\fi} }%
669 〈latexrelease〉 \reserved@a}%
670 〈latexrelease〉\EndIncludeInRelease
671 〈*2ekernel〉

```

(End of definition for `\@iinput`.)

`\@input` Define `\@input` in terms of `\IfFileExists`. So this is a ‘safe input’ command, but the files input are not listed by `\listfiles`.

We don’t want `.aux`, `.toc` files etc be listed by `\listfiles`. However, something like `.bb1` probably should be listed and thus should be implemented not by `\@input`.

```

672 \def\@input#1{%
673 \IfFileExists{#1}{\@input\@filef@und}{\typeout{No file #1.}}}

```

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

`\@input@` Version of `\@input` that does add the file to `\@filelist`.

```

674 \def\@input@#1{\InputIfFileExists{#1}{}{\typeout{No file #1.}}}

```

(End of 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.

```

675 〈/2ekernel〉
676 〈*2ekernel | latexrelease〉
677 〈latexrelease〉\IncludeInRelease{2020/10/01}%
678 〈latexrelease〉 {\@missingfileerror}{Do not load missing file immediately}%
679 〈gdef\@missingfileerror#1#2{%
680 \typeout{^^J! LaTeX Error: File '#1.#2' not found.^^J^^J%
681 Type X to quit or <RETURN> to proceed,^^J%
682 or enter new name. (Default extension: #2)^^J}%
683 \message{Enter file name: }%
684 {\endlinechar\m@ne

```

```

685 \global\read\m@ne to\@gtempa}%
686 \ifx\@gtempa\@empty

```

If the user answers with *<return>*, fallback to the .tex file (previously it did nothing).

```

687 \let\@missingfile@area\@empty
688 \let\@missingfile@base\@empty
689 \def\@missingfile@ext{tex}%
690 \else

```

Use \batchmode\read-1 to *<tl>* to end the T<sub>E</sub>X run, same as expl3 does (it was \batchmode\@@end before).

```

691 \def\reserved@b{\batchmode\read-1 to \reserved@a}%
692 \def\reserved@a{x}\ifx\reserved@a\@gtempa\reserved@b\fi
693 \def\reserved@a{X}\ifx\reserved@a\@gtempa\reserved@b\fi
694 \filename@parse\@gtempa
695 \edef\filename@ext{%
696 \ifx\filename@ext\relax#2\else\filename@ext\fi}%
697 \edef\reserved@a{%

```

Only check \IfFileExists (it was \InputIfFileExists).

```

698 \noexpand\IfFileExists
699 {\filename@area\filename@base.\filename@ext}%

```

If the file exists, define \@missingfile@{part}.

```

700 {\def\noexpand\@missingfile@area{\filename@area}%
701 \def\noexpand\@missingfile@base{\filename@base}%
702 \def\noexpand\@missingfile@ext {\filename@ext}}%
703 {\noexpand\@missingfileerror
704 {\filename@area\filename@base}{\filename@ext}}%
705 \reserved@a
706 \fi
707 }
708 (/2ekernel | latexrelease)
709 (latexrelease)\EndIncludeInRelease
710 (latexrelease)\IncludeInRelease{0000/00/00}%
711 (latexrelease) {(\@missingfileerror){Do not load missing file immediately}}%
712 (latexrelease)
713 (latexrelease)\gdef\@missingfileerror#1#2{%
714 (latexrelease) \typeout{^^J! LaTeX Error: File '#1.#2' not found.^^J^^J%
715 (latexrelease) Type X to quit or <RETURN> to proceed,^^J%
716 (latexrelease) or enter new name. (Default extension: #2)^J}%
717 (latexrelease) \message{Enter file name: }%
718 (latexrelease) {\endlinechar\m@ne
719 (latexrelease) \global\read\m@ne to\@gtempa}%
720 (latexrelease) \ifx\@gtempa\@empty
721 (latexrelease) \else
722 (latexrelease) \def\reserved@a{x}\ifx\reserved@a\@gtempa\batchmode\@@end\fi
723 (latexrelease) \def\reserved@a{X}\ifx\reserved@a\@gtempa\batchmode\@@end\fi
724 (latexrelease) \filename@parse\@gtempa
725 (latexrelease) \edef\filename@ext{%
726 (latexrelease) \ifx\filename@ext\relax#2\else\filename@ext\fi}%
727 (latexrelease) \edef\reserved@a{%
728 (latexrelease) \noexpand\InputIfFileExists
729 (latexrelease) {\filename@area\filename@base.\filename@ext}}%
730 (latexrelease) {}}%

```

```

731 〈\latexrelease〉 {\noexpand\@missingfileerror
732 〈\latexrelease〉 {\filename@area\filename@base}\{\filename@ext\}}}}%
733 〈\latexrelease〉 \reserved@a
734 〈\latexrelease〉 \fi}
735 〈\latexrelease〉
736 〈\latexrelease〉\EndIncludeInRelease
737 {*2ekernel}

(End of definition for \@missingfileerror.)

```

**\@obsoletefile** For compatibility with L<sup>A</sup>T<sub>E</sub>X 2.09 document styles, we distribute files called `article.sty`, `book.sty`, `report.sty`, `slides.sty` and `letter.sty`. These use the command `\@obsoletefile`, which produces a warning message.

```

738 \def\@obsoletefile#1#2{%
739 \@latex@warning@no@line{inputting '#1' instead of obsolete '#2'}}
740 \onlypreamble\@obsoletefile

```

## 1.2 Listing files

**\@filelist** A list of files input so far. The initial value of `\@gobble` eats the comma before the first file name.

```
741 \let\@filelist\@gobble
```

**\@addtofilelist** 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.

```
742 \%def\@addtofilelist#1{\xdef\@filelist{\@filelist,#1}}
```

A preamble command to cause `\end{document}` to list files input from the main file.

```

\listfiles
743 \def\listfiles{%
744 \let\listfiles\relax
745 \def\@listfiles##1##2##3##4##5##6##7##8##9@@{%
746 \def\reserved@d{\{}%
747 \atfor\reserved@c:={##1##2##3##4##5##6##7##8}\do{%
748 \ifx\reserved@c\reserved@d
749 \edef\filename@area{\filename@area}%
750 \fi}%
751 \def\@dofilelist{%
752 \typeout{^^J *File List*}%
753 \@for\@currname:=\@filelist\do{%
754 \filename@parse\@currname
755 \edef\reserved@a{%
756 \filename@base.%}
757 \ifx\filename@ext\relax tex\else\filename@ext\fi}%
758 \expandafter\let\expandafter\reserved@b
759 \csname ver@\reserved@a\endcsname

```

Packages that `\relax` their `\ver@...` string to allow for multiple loading (e.g., `fontenc`) can use `\ver@...` to store the version information instead.

```

760 \ifx\reserved@b\relax
761 \expandafter\let\expandafter\reserved@b
762 \csname ver@\@reserved@a\endcsname
763 \fi
764 \expandafter\expandafter\expandafter\@listfiles\expandafter

```

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}
 770 \onlypreamble\listfiles
```

```
\@dofilelist 771 \let\@dofilelist\relax
 772 \langle /2ekernel\rangle
(End of definition for \obsoletefile and others.)
```

# File s

## ltoutenc.dtx

### 1 Font encodings

This section of the kernel contains commands for declaring encoding-specific commands, such as accents. It also contains the code for some of the encoding files, including `omlenc.def`, `omsenc.def`, `t1enc.def` and `ot1enc.def` files, which define the OML, OMS, T1 and OT1 encodings, and the `fontenc` package for selecting encodings.

The `fontenc` package has options for encodings, of which the last option is the default encoding. For example, to use the OT2, OT3 and T1 encodings, with T1 as the default, you say:

```
\usepackage[OT2,OT3,T1]{fontenc}
```

The standard kernel set-up loads font encoding files and selects an encoding as follows.

```
\input {omlenc.def}
\input {t1enc.def}
\input {ot1enc.def}
\input {omsenc.def}
\fontencoding{OT1}
```

Note that the files in the standard `inputenc` package depend on this behaviour of the kernel.

The syntax for declaring encoding-specific commands is:

```
\DeclareTextCommand{\command}{{\encoding}}
[{\number} [{\default}]] {{\commands}}
```

This command is like `\newcommand`, except that it defines a command which is specific to one encoding. The resulting command is always robust, even if its definition is fragile. For example, the definition of `\l` in the OT1 encoding is:

```
\DeclareTextCommand{\l}{OT1}{{\@xxxii 1}}
```

`\DeclareTextCommand` takes the same optional arguments as `\newcommand`.

```
\ProvideTextCommand{\command}{{\encoding}}
[{\number} [{\default}]] {{\commands}}
```

This acts like `\DeclareTextCommand`, but does nothing if the command is already defined.

```
\DeclareTextSymbol{\command}{{\encoding}}{\slot}
```

This command defines a text symbol, with a particular slot in that encoding. The commands:

```
\DeclareTextSymbol{\ss}{OT1}{25}
\DeclareTextCommand{\ss}{OT1}{\char25 }
```

have the same effect, but the `\DeclareTextSymbol` is faster.

```
\DeclareTextAccent{\command}{{\encoding}}{\slot}
```

This command declares a text accent. The commands:

```
\DeclareTextAccent{"}{OT1}{127}
\DeclareTextCommand{"}{OT1}{\add@accent {127}}
```

have the same effect.

```
\DeclareTextComposite{<command>}
 {<encoding>}{<argument>}{<slot>}
```

This command declares a composite letter, for example in the T1 encoding '\{a} is slot 225, which is declared by:

```
\DeclareTextComposite{'}{T1}{a}{225}
```

The *command* will normally have been declared with `\DeclareTextAccent`, or as a one-argument `\DeclareTextCommand`.

`\DeclareTextComposite` is the most common example of using the more general declaration `\DeclareTextCompositeCommand`, which can define a composite to be an arbitrary piece of text.

```
\DeclareTextCompositeCommand{<command>}
 {<encoding>}{<argument>}{<text>}
```

For example, in the OT1 encoding Å has a hand-crafted definition this is declared as follows

```
\DeclareTextCompositeCommand{r}{OT1}{A}
 {\leavevmode\setbox\z@\hbox{!}\dimen@\ht\z@\advance\dimen@-1ex%
 \rlap{\raise.67\dimen\hbox{\char23}}A}
```

The *command* will normally have been declared with `\DeclareTextAccent`, or as a one-argument `\DeclareTextCommand`.

The commands defined using the above declarations can be used in two ways. Normally they are used by just calling the command in the appropriate encoding, for example `\ss`. However, sometimes you may wish to use a command in an encoding where it is not defined. If the command has no arguments, then you can use it in another encoding by calling `\UseTextSymbol`:

```
\UseTextSymbol{<encoding>}{<command>}
```

For example, `\UseTextSymbol{OT1}{\ss}` has the same effect as:

```
{\fontencoding{OT1}\selectfont\ss}
```

If the command has one argument then you can use it in another encoding by calling `\UseTextAccent`:

```
\UseTextAccent{<encoding>}{<command>}{<text>}
```

For example, if the current encoding is OT2 then `\UseTextAccent{OT1}{'}{a}` has the same effect as:

```
{\fontencoding{OT1}\selectfont'\{\fontencoding{OT2}\selectfont a\}}
```

You can also declare a default definition for a text command, which will be used if the current encoding has no appropriate definition. Such use will also set the definition for this command in the current encoding to equal this default definition; this makes subsequent uses of the command much faster.

```
\DeclareTextCommandDefault{\command}{\definition}
```

For example, the default definition of the command `\textonequarter` (which produces the fraction  $\frac{1}{4}$ ) could be built using math mode:

```
\DeclareTextCommandDefault{\textonequarter}{\ensuremath {\frac{1}{4}}}
```

There is a matching `\Provide` command which will not override an existing default definition:

```
\ProvideTextCommandDefault{\command}{\definition}
```

The most common use for these commands is to use symbols from other encodings, so there are some optimizations provided:

```
\DeclareTextSymbolDefault{\command}{\encoding}
\DeclareTextAccentDefault{\command}{\encoding}
```

are short for:

```
\DeclareTextCommandDefault{\command}
 {\UseTextSymbol{\encoding}{\command}}
\DeclareTextCommandDefault[1]{\command}
 {\UseTextAccent{\encoding}{\command}{{#1}}}
```

For example, to make OT1 the default encoding for `\ss` and `\'` you say:

```
\DeclareTextSymbolDefault{\ss}{OT1}
\DeclareTextAccentDefault{\'}{OT1}
```

Note that you can use these commands on any zero- or one-argument commands declared with `\DeclareText*` or `\ProvideText*`, not just those defined using `\DeclareTextSymbol` or `\DeclareTextAccent`.

## 1.1 Removing encoding-specific commands

In some cases encoding definitions are given to provide some limited support since nothing better is available, for example, the definition for `\textdollar` in OT1 is a hack since \$ and £ actually share the same slot in this encoding. Thus if such a glyph becomes available in a different encoding (e.g., TS1) one would like to get rid of the flaky one and make the default definition point to the new encoding. In such a case defining

```
\DeclareTextSymbol{\textdollar}{TS1}{36}
\DeclareTextSymbolDefault{\textdollar}{TS1}
```

is not enough since if typesetting in OT1 L<sup>A</sup>T<sub>E</sub>X will still find the encoding specific definition for OT1 and therefore ignore the new default. Therefore to ensure that in this case the TS1 version is used we have to remove the OT1 declaration:

```
\UndeclareTextCommand{\textdollar}{OT1}
```

Since the \$ sign is a proper glyph in the T1 encoding there is no point removing its definition and forcing L<sup>A</sup>T<sub>E</sub>X to pick up the TS1 version if typesetting in this encoding. However, assume you want to use the variant dollar sign, i.e., \$ for your dollars. In that case you have to get rid of the T1 declaration as well, e.g., the following would do that for you:

```
\UndeclareTextCommand{\textdollar}{OT1}
\UndeclareTextCommand{\textdollar} {T1}
\DeclareTextCommandDefault{\textdollar}
 {\UseTextSymbol{TS1}\textdollaroldstyle}
```

## 1.2 The order of declarations

If an encoding-specific command is defined for more than one encoding, then it will execute fastest in the encoding in which it was defined last since its top-level definition will be set up to execute in that encoding without any overhead.

For this reason the file `fonttext.ltx` currently first loads the definitions for the T1 encoding and then those for the OT1 encoding so that typesetting in OT1 is optimized since that is (still) the default. However, when T1 is explicitly requested (via `\usepackage[T1]{fontenc}`) the top-level definitions are automatically changed to favour T1 since its declarations are reloaded in the process.

For the same reason default declarations should never come last since they are implemented as a special encoding themselves (with the name ?). Specifying them last would simply mean to make those encoding-specific commands equally inefficient in all encodings. Therefore the `textcomp` package, for example, first sets up all defaults to point to TS1 and then declares the commands in the TS1 encoding.

## 1.3 Docstrip modules

This .dtx file is be used to generate several related files containing font encoding definitions. The mutually exclusive docstrip options are listed here.

|          |                                                                                                                                                                                                 |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| T1       | generates <code>t1enc.def</code> for the Cork encoding.                                                                                                                                         |
| TS1      | generates <code>ts1enc.def</code> for the Text Companion encoding.                                                                                                                              |
| TS1sty   | generates <code>textcomp.sty</code> , package that sets up use of the Text Companion encoding.                                                                                                  |
| OT1      | generates <code>ot1enc.def</code> for Knuth's CM encoding.                                                                                                                                      |
| OMS      | generates <code>omsenc.def</code> for Knuth's math symbol encoding.                                                                                                                             |
| OML      | generates <code>omlenc.def</code> for Knuth's math letters encoding.                                                                                                                            |
| OT4      | generates <code>ot4enc.def</code> for the Polish extension to the OT1 encoding, created by B. Jackowski and M. Ry  ko for use with the Polish version of Computer Modern and Computer Concrete. |
| TU       | generates <code>tuenc.def</code> for Unicode font encoding.                                                                                                                                     |
| package  | generates <code>fontenc.sty</code> for selecting encodings.                                                                                                                                     |
| 2ekernel | for the kernel commands.                                                                                                                                                                        |

## 1.4 Definitions for the kernel

### 1.4.1 Declaration commands

This section contains definitions for commands such as accents which depend on the current encoding. These commands will usually be kept in `.def` files, for example `ot1enc.def` contains the definitions for the OT1 encoding.

```
1 {*2ekernel}
2 \message{font encodings,}
Far too many macros in one block here!
```

```
\DeclareTextCommand
\ProvideTextCommand
\DeclareTextSymbol
 @dec@text@cmd
\chardef@text@cmd
 @changed@cmd
 @changed@x
\TextSymbolUnavailable
 @inmathwarn
```

If you say:

```
\DeclareTextCommand{\foo}{T1}...
```

then `\foo` is defined to be `\T1-cmd \foo \T1\foo`, where `\T1\foo` is *one* control sequence, not two! We then call `\newcommand` to define `\T1\foo`.

```
3 \def\DeclareTextCommand{%
4 @dec@text@cmd\newcommand}
5 \def\ProvideTextCommand{%
6 @dec@text@cmd\providecommand}
7 \def@dec@text@cmd#1#2#3{%
8 \expandafter\def\expandafter#2%
9 \expandafter{%
10 \csname#3-cmd\expandafter\endcsname
11 \expandafter#2%
12 \csname#3\string#2\endcsname
13 }%
14 \let@\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 of 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 of 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 of 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 of definition for `\add@accent`.)*

```
\hmode@bgroup
75 \def\hmode@bgroup{\leavevmode\bgroup}
```

*(End of 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{%
```

```

91 \def\expandafter\noexpand
92 \csname#2\string#\endcsname####1{%
93 \noexpand\@text@composite
94 \expandafter\noexpand\csname#2\string#\endcsname
95 ####1\noexpand\empty\noexpand\@text@composite
96 {##1}}}}%
97 \expandafter\reserved@c\expandafter{\reserved@a{##1}}%
98 \fi
99 \expandafter\def\csname\expandafter\string\csname
100 #2\endcsname\string#1-\string#3\empty\endcsname{#4}%
101 }
102 />2ekernel | latexrelease)
103 (latexrelease)\EndIncludeInRelease
104 (latexrelease)\IncludeInRelease{0000/00/00}{\DeclareTextCompositeCommand}
105 (latexrelease) {test for undeclared accent}%
106 (latexrelease)\def\DeclareTextCompositeCommand#1#2#3#4{%
107 (latexrelease) \expandafter\let\expandafter\reserved@a
108 (latexrelease) \csname#2\string#\endcsname
109 (latexrelease) \expandafter\expandafter\expandafter\ifx
110 (latexrelease) \expandafter@\car\reserved@a\relax\relax@nil
111 (latexrelease) \else\@text@composite\else
112 (latexrelease) \edef\reserved@b##1{%
113 (latexrelease) \def\expandafter\noexpand
114 (latexrelease) \csname#2\string#\endcsname####1{%
115 (latexrelease) \noexpand\@text@composite
116 (latexrelease) \expandafter\noexpand\csname#2\string#\endcsname
117 (latexrelease) ####1\noexpand\empty\noexpand\@text@composite
118 (latexrelease) {##1}}}}%
119 (latexrelease) \expandafter\reserved@c\expandafter{\reserved@a{##1}}%
120 (latexrelease) \fi
121 (latexrelease) \expandafter\def\csname\expandafter\string\csname
122 (latexrelease) #2\endcsname\string#1-\string#3\empty\endcsname{#4}%
123 (latexrelease)\EndIncludeInRelease
124 (*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\'{}-\@empty`, which ought to be `\relax`, and so all is well. If we didn't include the `\@empty`, then `\'{}{}` would expand to:

```
\csname \string \T1\` - \string \endcsname
```

so the `\endcsname` would be `\string`ed` and the whole of the rest of the document would be put inside the `\csname`. This would not be good.

```
126 \def\@text@composite#1#2#3\@text@composite{%
127 \expandafter\@text@composite@x
128 \csname\string#1-\string#2\endcsname}
```

Originally the `\@text@composite@x` macro had two arguments and if #1 was not `\relax` it was executed, otherwise #2 was executed. All this happened within the `\ifx` code so that neither #1 nor #2 could have picked up any additional arguments from the input stream. This has now been changed using the typical `\@firstoftwo / \@secondoftwo` coding. This way the final expansion will happen without any `\else` or `\fi` intervening in the case that we need to get a further token from the input stream.

```
129 \def\@text@composite@x#1{%
130 \ifx#1\relax
131 \expandafter\@secondoftwo
132 \else
133 \expandafter\@firstoftwo
134 \fi
135 #1}
```

The command `\DeclareTextComposite` uses `\DeclareTextCompositeCommand` to declare a command which expands out to a single glyph.

```
136 \catcode\z@=11\relax
137 \def\DeclareTextComposite#1#2#3#4{%
138 \def\reserved@a{\DeclareTextCompositeCommand#1{#2}{#3}}%
139 \bgroup
140 \lccode\z@#4%
141 \lowercase{%
142 \egroup
143 \reserved@a ^^@}}
144 \catcode\z@=15\relax
145 \onlypreamble\DeclareTextComposite
```

*(End of 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 of 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 of definition for \hmode@start@before@group.)

\DeclareTextSymbolDefault Some syntactic sugar. Again, these should probably be optimized for speed.

```
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
```

(End of definition for \DeclareTextSymbolDefault and \DeclareTextAccentDefault.)

\UndeclareTextCommand 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.

```
191 \def\UndeclareTextCommand#1#2{%
```

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).

```
192 \expandafter\ifx\csname#2\string#1\endcsname\relax
193 \else
```

Else: throw away that declaration.

```
194 \global\expandafter\let\csname#2\string#1\endcsname
195 \undefined
```

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 \foo would look similar to \T1-cmd \foo \T1\foo (three tokens).

Of course, instead of T1 one could see a different encoding name; which one depends the encoding for which \foo was declared last.

Now assume we have just removed the declaration for \foo in T1 and the top-level of \foo expands to the above. Then we better change that pretty fast otherwise we do get an “undefined csname error” when we try to typeset \foo within T1 instead of getting the default definition for \foo. 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 ?.

Thus in case the last token of the top-level expansion is now undefined we change the declaration to look like \?-cmd \foo \?\foo which is done by the following (readable?) code:

```
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 }
204 \onlypreamble\UndeclareTextCommand
```

(End of definition for \UndeclareTextCommand.)

### 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 of 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 of definition for `\a`.)

### 1.4.4 Default encodings

We define the default encodings for most commands to be either OT1, OML or OMS. These defaults are in the kernel and therefore fonts with these encodings must be available unless these defaults are redefined elsewhere. Recall that the standard kernel loads the encoding files for these encodings, and also that for the T1 encoding.

The naming conventions in the kernel are not what we would use if we were starting from scratch... Those defined by DEK (like `\ae` and `\ss`) or by the TeX Users Group Technical Working Group on multi-lingual typesetting (like `\th` and `\ng`) have short names. Those which were added to the kernel in 1993 and early 1994 are named after their Adobe glyph names (like `\guillemotleft` and `\quotedblbase`). Unfortunately, this naming scheme won't work for all glyphs, since some names (like `\space`) are already used, and some (like `\endash`) are very likely to be defined by users. So we're now using the naming scheme of `\text` followed by the Adobe name, (like `\textendash` and `\textsterling`). Except that some glyphs don't have Adobe names, so we're using the names used by fontinst for those (like `\textcompwordmark`). Sigh.

Some accents from OT1:

```
226 \DeclareTextAccentDefault{\\"}{OT1}
227 \DeclareTextAccentDefault{\'}{OT1}
228 \DeclareTextAccentDefault{\.{}}{OT1}
229 \DeclareTextAccentDefault{\=}{OT1}
230 \DeclareTextAccentDefault{\H}{OT1}
231 \DeclareTextAccentDefault{\^}{OT1}
232 \DeclareTextAccentDefault{\`}{OT1}
233 \DeclareTextAccentDefault{\b}{OT1}
234 \DeclareTextAccentDefault{\c}{OT1}
235 \DeclareTextAccentDefault{\d}{OT1}
236 \DeclareTextAccentDefault{\r}{OT1}
237 \DeclareTextAccentDefault{\u}{OT1}
238 \DeclareTextAccentDefault{\v}{OT1}
239 \DeclareTextAccentDefault{\~}{OT1}
```

Some symbols from OT1:

```
240 \% \DeclareTextSymbolDefault{\AA}{OT1}
241 \DeclareTextSymbolDefault{\AE}{OT1}
242 \DeclareTextSymbolDefault{\L}{OT1}
243 \DeclareTextSymbolDefault{\OE}{OT1}
244 \DeclareTextSymbolDefault{\O}{OT1}
245 \% \DeclareTextSymbolDefault{\aa}{OT1}
246 \DeclareTextSymbolDefault{\ae}{OT1}
247 \DeclareTextSymbolDefault{\i}{OT1}
248 \DeclareTextSymbolDefault{\j}{OT1}

249 \DeclareTextSymbolDefault{\ij}{OT1}
250 \DeclareTextSymbolDefault{\IJ}{OT1}

251 \DeclareTextSymbolDefault{\l}{OT1}
252 \DeclareTextSymbolDefault{\oe}{OT1}
253 \DeclareTextSymbolDefault{\o}{OT1}
254 \DeclareTextSymbolDefault{\ss}{OT1}
255 \DeclareTextSymbolDefault{\textdollar}{OT1}
256 \DeclareTextSymbolDefault{\textemdash}{OT1}
257 \DeclareTextSymbolDefault{\textendash}{OT1}
258 \DeclareTextSymbolDefault{\textexclamdown}{OT1}
259 \% \DeclareTextSymbolDefault{\texthphenchar}{OT1}
260 \% \DeclareTextSymbolDefault{\texthphen}{OT1}
261 \DeclareTextSymbolDefault{\textquestiondown}{OT1}
262 \DeclareTextSymbolDefault{\textquotedblleft}{OT1}
263 \DeclareTextSymbolDefault{\textquotedblright}{OT1}
264 \DeclareTextSymbolDefault{\textquotleft}{OT1}
265 \DeclareTextSymbolDefault{\textquotright}{OT1}
266 \DeclareTextSymbolDefault{\textsterling}{OT1}
```

Some symbols from OMS:

```
267 \DeclareTextSymbolDefault{\textasteriskcentered}{OMS}
268 \DeclareTextSymbolDefault{\textbackslash}{OMS}
269 \DeclareTextSymbolDefault{\textbar}{OMS}
270 \DeclareTextSymbolDefault{\textbardbl}{OMS}
271 \DeclareTextSymbolDefault{\textbraceleft}{OMS}
272 \DeclareTextSymbolDefault{\textbraceright}{OMS}
273 \DeclareTextSymbolDefault{\textbullet}{OMS}
```

```

274 \DeclareTextSymbolDefault{\textdaggerdbl}{OMS}
275 \DeclareTextSymbolDefault{\textdagger}{OMS}
276 \DeclareTextSymbolDefault{\textparagraph}{OMS}
277 \DeclareTextSymbolDefault{\textperiodcentered}{OMS}
278 \DeclareTextSymbolDefault{\textsection}{OMS}
279 \DeclareTextAccentDefault{\textcircled}{OMS}

 Some symbols from OML:

280 \DeclareTextSymbolDefault{\textless}{OML}
281 \DeclareTextSymbolDefault{\textgreater}{OML}
282 \DeclareTextAccentDefault{\t}{OML}

 Some defaults we can fake.

 The interface for defining \copyright changed, it used to use \expandafter to add
braces at the appropriate points.

283 \DeclareTextCommandDefault{\textcopyright}{\textcircled{c}}
284 % \expandafter\def\expandafter
285 % \copyright\expandafter{\expandafter{\copyright}}
286 \DeclareTextCommandDefault{\textasciicircum}{\^{}}
287 \DeclareTextCommandDefault{\textasciitilde}{\~{}}
288 \DeclareTextCommandDefault{\textunderscore}{%
289 \leavevmode \kern.06em\vbox{\hrule\@width.3em}}}

 There is no good reason anymore to fake \textcompwordmark.

290 \% \DeclareTextCommandDefault{\textcompwordmark}{\leavevmode\kern\z@}
291 \DeclareTextSymbolDefault{\textcompwordmark}{T1}

292 \DeclareTextCommandDefault{\textvisiblespace}{%
293 \mbox{\kern.06em\vrule\@height.3ex}%
294 \vbox{\hrule\@width.3em}%
295 \hbox{\vrule\@height.3ex}%
}

 Using \fontdimen3 in the next definition is some sort of a kludge (since it is the
interword stretch) but it makes the ellipsis come out right in mono-spaced fonts too (since
there it is zero).

296 \DeclareTextCommandDefault{\textellipsis}{%
297 .\kern\fontdimen3\font
298 .\kern\fontdimen3\font
299 .\kern\fontdimen3\font}

300 \% \DeclareTextCommandDefault{\textregistered}{\textcircled{\scshape r}}
301 \DeclareTextCommandDefault{\textregistered}{\textcircled{%
302 \check@mathfonts\fontsize\sf@size\z@\math@fontsfalse\selectfont R}}
303 \DeclareTextCommandDefault{\texttrademark}{TM}
304 \DeclareTextCommandDefault{\SS}{SS}

305 \DeclareTextCommandDefault{\textordfeminine}{a}
306 \DeclareTextCommandDefault{\textordmasculine}{o}

```

### 1.4.5 Math material

Some commands can be used in both text and math mode:

```
307 \DeclareRobustCommand{\$}{\ifmmode\mathdollar\else\textrm{\$}\fi}
```

We use `\protected` not `\DeclareRobustCommand` so that `\bigl\{` etc. works inside `\protected@edef`.

```
308 \protected\def{\{\ifmmode\lbrace\else\textrm{\{}fi\}}
309 \protected\def{\}\ifmmode\rbrace\else\textrm{\}}fi\}
310 \DeclareRobustCommand{\P}{\ifmmode\mathparagraph\else\textrm{\P}\fi}
311 \DeclareRobustCommand{\S}{\ifmmode\mathsection\else\textrm{\S}\fi}
312 \DeclareRobustCommand{\dag}{\ifmmode\dagger\else\textrm{\dag}\fi}
313 \DeclareRobustCommand{\ddag}{\ifmmode\ddagger\else\textrm{\ddagger}\fi}
```

For historical reasons `\copyright` needs {} around the definition in maths.

```
314 \DeclareRobustCommand{_}{%
315 \ifmmode\nfss@text{\textunderscore}\else\textrm{_}\fi}
316 \DeclareRobustCommand{\copyright}{%
317 \ifmmode{\nfss@text{\textcopyright}}\else\textrm{\copyright}\fi}
318 \DeclareRobustCommand{\pounds}{%
319 \ifmmode\mathsterling\else\textrm{\pounds}\fi}
320 \DeclareRobustCommand{\dots}{%
321 \ifmmode\mathellipsis\else\textrm{\dots}\fi}
322 \let\ldots\dots
```

Default definition of the commabelow accent.

```
323 </2ekernel>
324 <texrel>\IncludeInRelease{2015/10/01}{\textcommabelow}{comma accent}%
325 <2ekernel | texrel>
326 \DeclareTextCommandDefault{\textcommabelow}[1]
327 {\hmode@bgroup\oalign{\null#1\crcr\hidewidth\raise-.31ex
328 \hbox{\check@mathfonts\fontsize\ssf@size\z@
329 \math@fontsfalse\selectfont,\hidewidth}\egroup}
330 <texrel>\EndIncludeInRelease
331 <2ekernel | texrel>
332 <texrel>\IncludeInRelease{0000/00/00}{\textcommabelow}{comma accent}%
333 <texrel>\let\textcommabelow\@undefined
334 <texrel>\expandafter
335 <texrel> \let\csname\string\T1\string\c-G\endcsname\@undefined
336 <texrel>\expandafter
337 <texrel> \let\csname\string\T1\string\c-K\endcsname\@undefined
338 <texrel>\expandafter
339 <texrel> \let\csname\string\T1\string\c-k\endcsname\@undefined
340 <texrel>\expandafter
341 <texrel> \let\csname\string\T1\string\c-L\endcsname\@undefined
342 <texrel>\expandafter
343 <texrel> \let\csname\string\T1\string\c-l\endcsname\@undefined
344 <texrel>\expandafter
345 <texrel> \let\csname\string\T1\string\c-N\endcsname\@undefined
346 <texrel>\expandafter
347 <texrel> \let\csname\string\T1\string\c-n\endcsname\@undefined
348 <texrel>\expandafter
349 <texrel> \let\csname\string\T1\string\c-R\endcsname\@undefined
```

```

350 \let\csname\string\T1\string\c-r\endcsname\@undefined
351 \let\csname\string\T1\string\c-r\endcsname\@undefined
352 \EndIncludeInRelease
 Default definition of the commaabove accent(E.G.).
353 \IncludeInRelease{2016/02/01}{\textcommablock}{comma above}%
354 {*2ekernel | latexrelease}
355 \DeclareTextCommandDefault{\textcommablock}[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}{\textcommablock}{comma above}%
370 \let\textcommablock\@undefined
371 \expandafter
372 \let\csname\string\OT1\string\c-g\endcsname\@undefined
373 \expandafter
374 \let\csname\string\T1\string\c-g\endcsname\@undefined
375 \EndIncludeInRelease

```

## 1.5 Definitions for the OT1 encoding

The definitions for the ‘TEX text’ (OT1) encoding.

Declare the encoding.

```

376 {*OT1}
377 \DeclareFontEncoding{OT1}{}{}

```

Declare the accents.

```

378 \DeclareTextAccent{"}{OT1}{127}
379 \DeclareTextAccent{'}{OT1}{19}
380 \DeclareTextAccent{.}{OT1}{95}
381 \DeclareTextAccent{=}{OT1}{22}
382 \DeclareTextAccent{^}{OT1}{94}
383 \DeclareTextAccent{`}{OT1}{18}
384 \DeclareTextAccent{~-}{OT1}{126}
385 \DeclareTextAccent{H}{OT1}{125}
386 \DeclareTextAccent{u}{OT1}{21}
387 \DeclareTextAccent{v}{OT1}{20}
388 \DeclareTextAccent{r}{OT1}{23}

```

Some accents have to be built by hand: Note that `\oalign` and `\o@align` must be inside a group. In these definitions we no longer use the helper function `\sh@ft` from plain.tex since that now has two incompatible definitions.

```

389 \DeclareTextCommand{\b}{OT1}[1]
390 {\hmode@bgroup\o@align{\relax#1\crcr\hidewidth\ltx@sh@ft{-3ex}}%

```

```

391 \vbox to .2ex{\hbox{\char22}\vss}\hidewidth}\egroup}
392 \DeclareTextCommand{\c}{OT1}[1]
393 {\leavevmode\setbox\z@\hbox{\#1}\ifdim\ht\z@=1ex\accent24 #1%
394 \else{\ooalign{\unhbox\z@\crcr\hidewidth\char24\hidewidth}}\fi}
395 \DeclareTextCommand{\d}{OT1}[1]
396 {\hmode@bgroup
397 \o@lign{\relax#1\crcr\hidewidth\ltx@sh@ft{-1ex}. \hidewidth}\egroup}

```

Declare the text symbols.

```

398 \DeclareTextSymbol{\AE}{OT1}{29}
399 \DeclareTextSymbol{\OE}{OT1}{30}
400 \DeclareTextSymbol{\O}{OT1}{31}
401 \DeclareTextSymbol{\ae}{OT1}{26}
402 \DeclareTextSymbol{\i}{OT1}{16}
403 \DeclareTextSymbol{\j}{OT1}{17}
404 \DeclareTextSymbol{\oe}{OT1}{27}
405 \DeclareTextSymbol{\o}{OT1}{28}
406 \DeclareTextSymbol{\ss}{OT1}{25}
407 \DeclareTextSymbol{\textemdash}{OT1}{124}
408 \DeclareTextSymbol{\textendash}{OT1}{123}

```

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

```

409 \DeclareTextCommand{\textnonbreakinghyphen}{OT1}{\mbox{-}\nobreak\hspace{z@}}
410 \DeclareTextCommand{\textfiguredash} {OT1}{\textendash}
411 \DeclareTextCommand{\texthorizontalbar} {OT1}{\textemdash}

```

Using the ligatures helps with OT1 fonts that have `\textexclamdown` and `\textquestiondown` in unusual positions.

```

412 %\DeclareTextSymbol{\textexclamdown}{OT1}{60}
413 %\DeclareTextSymbol{\textquestiondown}{OT1}{62}
414 \DeclareTextCommand{\textexclamdown}{OT1}{!`}
415 \DeclareTextCommand{\textquestiondown}{OT1}{?`}
416 %\DeclareTextSymbol{\texthyphenchar}{OT1}{`-}
417 %\DeclareTextSymbol{\texthyphen}{OT1}{`-}
418 \DeclareTextSymbol{\textquotedblleft}{OT1}{92}
419 \DeclareTextSymbol{\textquotedblright}{OT1}{`}
420 \DeclareTextSymbol{\textquotelleft}{OT1}{`}
421 \DeclareTextSymbol{\textquoteright}{OT1}{`}

```

Some symbols which are faked from others:

```

422 % \DeclareTextCommand{\aa}{OT1}
423 % {{\accent23a}}
424 \DeclareTextCommand{\L}{OT1}
425 {\leavevmode\setbox\z@\hbox{L}\hb@xt@wd\z@{\hss@xxxii L}}
426 \DeclareTextCommand{\l}{OT1}
427 {\hmode@bgroup\@xxxii l\egroup}
428 % \DeclareTextCommand{\AA}{OT1}
429 % {\leavevmode\setbox\z@\hbox{h}\dimen@ht\z@\advance\dimen@-1ex%
430 % \rlap{\raise.67\dimen@\hbox{\char23}}A}

```

In the OT1 encoding Å has a hand-crafted definition, so we have here the first recorded explicit use of `\DeclareTextCompositeCommand`.

```

431 \DeclareTextCompositeCommand{\r}{OT1}{A}
432 {\leavevmode\setbox\z@\hbox{!}\dimen@ht\z@\advance\dimen@-1ex%
433 \rlap{\raise.67\dimen@\hbox{\char23}}A}

```

The dutch language uses the letter ‘ij’. It is available in T1 encoded fonts, but not in the OT1 encoded fonts. Therefore we fake it for the OT1 encoding.

```

434 \DeclareTextCommand{\ij}{OT1}{%
435 \nobreak\hskip\z@skip i\kern-0.02em\nobreak\hskip\z@skip j}
436 \DeclareTextCommand{\IJ}{OT1}{%
437 \nobreak\hskip\z@skip I\kern-0.02em\nobreak\hskip\z@skip J}

```

In the OT1 encoding, £ and \$ share a slot.

```

438 \DeclareTextCommand{\textdollar}{OT1}{\hmode@bgroup
439 \ifdim \fontdimen\@ne\font >\z@
440 \slshape
441 \else
442 \upshape
443 \fi
444 \char`\$\egroup}
445 \DeclareTextCommand{\textsterling}{OT1}{\hmode@bgroup
446 \ifdim \fontdimen\@ne\font >\z@
447 \itshape
448 \else
449 \fontshape{ui}\selectfont
450 \fi
451 \char`\$\egroup}

```

Here we are adding some more composite commands to the OT1 encoding. This makes the use of certain accents with i compatible with their use with the T1 encoding; this enables them to become true L<sup>A</sup>T<sub>E</sub>X internal representations. However, it will make these accents work a little less fast since a check will always be made for the existence of a composite.

```

452 \DeclareTextComposite{\.}{OT1}{i}{`i}
453 \DeclareTextComposite{\.}{OT1}{i}{`i}
454 \DeclareTextCompositeCommand{\`}{OT1}{i}{\@tabacckludge`i}
455 \DeclareTextCompositeCommand{\'}{OT1}{i}{\@tabacckludge'i}
456 \DeclareTextCompositeCommand{\^}{OT1}{i}{^\i}
457 \DeclareTextCompositeCommand{\"}{OT1}{i}{\"i}

```

T1 encoding is given more extensive set of overloads for \c. But here we just adjust \c{g}.

```

458 \ifx\textcommaabove\undefined\else
459 \DeclareTextCompositeCommand{\c}{OT1}{g}{\textcommaabove{g}}
460 \fi
461
```

## 1.6 Definitions for the T1 encoding

The definitions for the ‘Extended T<sub>E</sub>X text’ (T1) encoding.

Declare the encoding.

```

462 <*T1>
463 \DeclareFontEncoding{T1}{}{}

```

Declare the accents.

```

464 \DeclareTextAccent{\`}{T1}{0}
465 \DeclareTextAccent{\'}{T1}{1}
466 \DeclareTextAccent{\^}{T1}{2}

```

```

467 \DeclareTextAccent{\~}{T1}{3}
468 \DeclareTextAccent{\^}{T1}{4}
469 \DeclareTextAccent{\H}{T1}{5}
470 \DeclareTextAccent{\r}{T1}{6}
471 \DeclareTextAccent{\v}{T1}{7}
472 \DeclareTextAccent{\u}{T1}{8}
473 \DeclareTextAccent{\=}{T1}{9}
474 \DeclareTextAccent{\.}{T1}{10}

```

Some accents have to be built by hand. Note that `\o@align` and `\o@lign` must be inside a group. In these definitions we no longer use the helper function `\sh@ft` from plain.tex since that now has two incompatible definitions.

```

475 \DeclareTextCommand{\b}{T1}[1]
476 {\hmode@bgroup\o@lign{\relax#1\crcr\hidewidth\ltx@sh@ft{-3ex}%
477 \vbox to .2ex{\hbox{\char9}\vss}\hidewidth}\egroup}
478 \DeclareTextCommand{\c}{T1}[1]
479 {\leavevmode\setbox\z@\hbox{\#1}\ifdim\ht\z@=1ex\accent11 #1%
480 \else\o@align{\unhbox\z@\crcr
481 \hidewidth\char11\hidewidth}\fi}
482 \DeclareTextCommand{\d}{T1}[1]
483 {\hmode@bgroup
484 \o@lign{\relax#1\crcr\hidewidth\ltx@sh@ft{-1ex}.\hidewidth}\egroup}
485 \DeclareTextCommand{\k}{T1}[1]
486 {\hmode@bgroup\o@align{\null\#1\crcr\hidewidth\char12}\egroup}
487 \DeclareTextCommand{\textogonekcentered}{T1}[1]
488 {\hmode@bgroup\o@align{%
489 \null\#1\crcr\hidewidth\char12\hidewidth}\egroup}

```

Some symbols are constructed.

Slot 24 contains a small circle intended for construction of these two glyphs.

```

490 \DeclareTextCommand{\textperthousand}{T1}
491 {\%\char 24 } % space or 'relax as delimiter?
492 \DeclareTextCommand{\textpertenthousand}{T1}
493 {\%\char 24\char 24 } % space or 'relax as delimiter?

```

For Maltese, `\Hwithstroke` and `\hwithstroke` are needed.

```

494 \DeclareTextCommand{\Hwithstroke}{T1}
495 {%
496 \hmode@bgroup
497 \vphantom{H}%
498 \sbox\z@{H}%
499 \o@align{%
500 H\cr
501 \hidewidth
502 \vrule
503 height \dimexpr 0.7\ht\z@+0.1ex\relax
504 depth -0.7\ht\z@
505 width 0.8\wd\z@
506 \hidewidth\cr
507 }%
508 \egroup
509 }
510 \DeclareTextCommand{\hwithstroke}{T1}
511 {%

```

```

512 \hmode@bgroup
513 \vphantom{h}%
514 \sbox{z@{h}}%
515 \ooalign{%
516 h\cr
517 \kern0.075\wd{z@}
518 \vrule
519 height \dimexpr 0.7\ht{z@}+0.1ex\relax
520 depth -0.7\ht{z@}
521 width 0.4\wd{z@}
522 \hidewidth\cr
523 }%
524 \egroup
525 }
```

Declare the text symbols.

```

526 \%{\ DeclareTextSymbol{\AA}{T1}{197}
527 \ DeclareTextSymbol{\AE}{T1}{198}
528 \ DeclareTextSymbol{\DH}{T1}{208}
529 \ DeclareTextSymbol{\DJ}{T1}{208}
530 \ DeclareTextSymbol{\L}{T1}{138}
531 \ DeclareTextSymbol{\NG}{T1}{141}
532 \ DeclareTextSymbol{\OE}{T1}{215}
533 \ DeclareTextSymbol{\O}{T1}{216}
534 \ DeclareTextSymbol{\SS}{T1}{223}
535 \ DeclareTextSymbol{\TH}{T1}{222}
536 \%{\ DeclareTextSymbol{\aa}{T1}{229}
537 \ DeclareTextSymbol{\ae}{T1}{230}
538 \ DeclareTextSymbol{\dh}{T1}{240}
539 \ DeclareTextSymbol{\dj}{T1}{158}

540 \ DeclareTextSymbol{\guillemetleft}{T1}{19}
541 \ DeclareTextSymbol{\guillemetright}{T1}{20}
542 % old Adobe names
543 \ DeclareTextSymbol{\guillemotleft}{T1}{19}
544 \ DeclareTextSymbol{\guillemotright}{T1}{20}

545 \ DeclareTextSymbol{\guilsinglleft}{T1}{14}
546 \ DeclareTextSymbol{\guilsinglright}{T1}{15}
547 \ DeclareTextSymbol{\i}{T1}{25}
548 \ DeclareTextSymbol{\j}{T1}{26}
549 \ DeclareTextSymbol{\ij}{T1}{188}
550 \ DeclareTextSymbol{\IJ}{T1}{156}
551 \ DeclareTextSymbol{\l}{T1}{170}
552 \ DeclareTextSymbol{\ng}{T1}{173}
553 \ DeclareTextSymbol{\oe}{T1}{247}
554 \ DeclareTextSymbol{\o}{T1}{248}
555 \ DeclareTextSymbol{\quotedblbase}{T1}{18}
556 \ DeclareTextSymbol{\quotesinglbase}{T1}{13}
557 \ DeclareTextSymbol{\ss}{T1}{255}
558 \ DeclareTextSymbol{\textasciicircum}{T1}{`^}
559 \ DeclareTextSymbol{\textasciitilde}{T1}{`~}
560 \ DeclareTextSymbol{\textbackslash}{T1}{``\\`}
561 \ DeclareTextSymbol{\textbar}{T1}{`\|`}
562 \ DeclareTextSymbol{\textbraceleft}{T1}{`\{`}
```

```

563 \DeclareTextSymbol{\textbraceright}{T1}{`}
564 \DeclareTextSymbol{\textcompwordmark}{T1}{23}
565 \DeclareTextSymbol{\textdollar}{T1}{`}
566 \DeclareTextSymbol{\textemdash}{T1}{22}
567 \DeclareTextSymbol{\textendash}{T1}{21}

```

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

```

568 \DeclareTextCommand{\textnonbreakinghyphen}{T1}{\mbox{-}\nobreak\hskip\z@}
569 \DeclareTextCommand{\textfiguredash} {T1}{\textendash}
570 \DeclareTextCommand{\texthorizontalbar} {T1}{\textemdash}

571 \DeclareTextSymbol{\textexclamdown}{T1}{189}
572 \DeclareTextSymbol{\textgreater}{T1}{`}
573 %\DeclareTextSymbol{\texthyphenchar}{T1}{127}
574 %\DeclareTextSymbol{\texthyphen}{T1}{`}
575 \DeclareTextSymbol{\textless}{T1}{`}
576 \DeclareTextSymbol{\textquestiondown}{T1}{190}
577 \DeclareTextSymbol{\textquotedblleft}{T1}{16}
578 \DeclareTextSymbol{\textquotedblright}{T1}{17}
579 \DeclareTextSymbol{\textquotedbl}{T1}{`}
580 \DeclareTextSymbol{\textquotelleft}{T1}{`}
581 \DeclareTextSymbol{\textquoteright}{T1}{`}
582 \DeclareTextSymbol{\textsection}{T1}{159}
583 \DeclareTextSymbol{\textsterling}{T1}{191}
584 \DeclareTextSymbol{\textunderscore}{T1}{95}
585 \DeclareTextSymbol{\textvisiblespace}{T1}{32}
586 \DeclareTextSymbol{\th}{T1}{254}

```

Declare the composites.

```

587 \DeclareTextComposite{\.}{T1}{i}{`}
588 \DeclareTextComposite{\.}{T1}{i}{`}

```

"80 = 128

```

589 \DeclareTextComposite{\u}{T1}{A}{128}
590 \DeclareTextComposite{\k}{T1}{A}{129}
591 \DeclareTextComposite{\'}{T1}{C}{130}
592 \DeclareTextComposite{\v}{T1}{C}{131}
593 \DeclareTextComposite{\v}{T1}{D}{132}
594 \DeclareTextComposite{\v}{T1}{E}{133}
595 \DeclareTextComposite{\k}{T1}{E}{134}
596 \DeclareTextComposite{\u}{T1}{G}{135}

```

"88 = 136

```

597 \DeclareTextComposite{\'}{T1}{L}{136}
598 \DeclareTextComposite{\v}{T1}{L}{137}
599 \DeclareTextComposite{\'}{T1}{N}{139}
600 \DeclareTextComposite{\v}{T1}{N}{140}
601 \DeclareTextComposite{\H}{T1}{O}{142}
602 \DeclareTextComposite{\'}{T1}{R}{143}

```

"90 = 144

```

603 \DeclareTextComposite{\v}{T1}{R}{144}
604 \DeclareTextComposite{\'}{T1}{S}{145}
605 \DeclareTextComposite{\v}{T1}{S}{146}
606 \DeclareTextComposite{\c}{T1}{S}{147}

```

```

607 \DeclareTextComposite{\v}{T1}{T}{148}
608 \DeclareTextComposite{\c}{T1}{T}{149}
609 \DeclareTextComposite{\H}{T1}{U}{150}
610 \DeclareTextComposite{\r}{T1}{U}{151}

"A0 = 152
611 \DeclareTextComposite{\"}{T1}{Y}{152}
612 \DeclareTextComposite{\'}{T1}{Z}{153}
613 \DeclareTextComposite{\v}{T1}{Z}{154}
614 \DeclareTextComposite{\.}{T1}{Z}{155}
615 \DeclareTextComposite{\.}{T1}{I}{157}

"A0 = 160
616 \DeclareTextComposite{\u}{T1}{a}{160}
617 \DeclareTextComposite{\k}{T1}{a}{161}
618 \DeclareTextComposite{\'}{T1}{c}{162}
619 \DeclareTextComposite{\v}{T1}{c}{163}
620 \DeclareTextComposite{\v}{T1}{d}{164}
621 \DeclareTextComposite{\v}{T1}{e}{165}
622 \DeclareTextComposite{\k}{T1}{e}{166}
623 \DeclareTextComposite{\u}{T1}{g}{167}

"A8 = 168
624 \DeclareTextComposite{\'}{T1}{l}{168}
625 \DeclareTextComposite{\v}{T1}{l}{169}
626 \DeclareTextComposite{\'}{T1}{n}{171}
627 \DeclareTextComposite{\v}{T1}{n}{172}
628 \DeclareTextComposite{\H}{T1}{o}{174}
629 \DeclareTextComposite{\'}{T1}{r}{175}

"B0 = 176
630 \DeclareTextComposite{\v}{T1}{r}{176}
631 \DeclareTextComposite{\'}{T1}{s}{177}
632 \DeclareTextComposite{\v}{T1}{s}{178}
633 \DeclareTextComposite{\c}{T1}{s}{179}
634 \DeclareTextComposite{\v}{T1}{t}{180}
635 \DeclareTextComposite{\c}{T1}{t}{181}
636 \DeclareTextComposite{\H}{T1}{u}{182}
637 \DeclareTextComposite{\r}{T1}{u}{183}

"B8 = 184
638 \DeclareTextComposite{\"}{T1}{y}{184}
639 \DeclareTextComposite{\'}{T1}{z}{185}
640 \DeclareTextComposite{\v}{T1}{z}{186}
641 \DeclareTextComposite{\.}{T1}{z}{187}

"C0 = 192
642 \DeclareTextComposite{\'}{T1}{A}{192}
643 \DeclareTextComposite{\'}{T1}{A}{193}
644 \DeclareTextComposite{\^}{T1}{A}{194}
645 \DeclareTextComposite{\~}{T1}{A}{195}
646 \DeclareTextComposite{\"}{T1}{A}{196}
647 \DeclareTextComposite{\r}{T1}{A}{197}
648 \DeclareTextComposite{\c}{T1}{C}{199}

```

```

"C8 = 200
649 \DeclareTextComposite{\`}{T1}{E}{200}
650 \DeclareTextComposite{\'}{T1}{E}{201}
651 \DeclareTextComposite{\^}{T1}{E}{202}
652 \DeclareTextComposite{\"}{T1}{E}{203}
653 \DeclareTextComposite{\`}{T1}{I}{204}
654 \DeclareTextComposite{\'}{T1}{I}{205}
655 \DeclareTextComposite{\^}{T1}{I}{206}
656 \DeclareTextComposite{\\"}{T1}{I}{207}

"D0 = 208
657 \DeclareTextComposite{\~}{T1}{N}{209}
658 \DeclareTextComposite{\'}{T1}{O}{210}
659 \DeclareTextComposite{\'}{T1}{O}{211}
660 \DeclareTextComposite{\^}{T1}{O}{212}
661 \DeclareTextComposite{\~}{T1}{O}{213}
662 \DeclareTextComposite{\\"}{T1}{O}{214}

"D8 = 216
663 \DeclareTextComposite{\`}{T1}{U}{217}
664 \DeclareTextComposite{\'}{T1}{U}{218}
665 \DeclareTextComposite{\^}{T1}{U}{219}
666 \DeclareTextComposite{\\"}{T1}{U}{220}
667 \DeclareTextComposite{\'}{T1}{Y}{221}

"E0 = 224
668 \DeclareTextComposite{\`}{T1}{a}{224}
669 \DeclareTextComposite{\'}{T1}{a}{225}
670 \DeclareTextComposite{\^}{T1}{a}{226}
671 \DeclareTextComposite{\~}{T1}{a}{227}
672 \DeclareTextComposite{\\"}{T1}{a}{228}
673 \DeclareTextComposite{\r}{T1}{a}{229}
674 \DeclareTextComposite{\c}{T1}{c}{231}

"E8 = 232
675 \DeclareTextComposite{\`}{T1}{e}{232}
676 \DeclareTextComposite{\'}{T1}{e}{233}
677 \DeclareTextComposite{\^}{T1}{e}{234}
678 \DeclareTextComposite{\\"}{T1}{e}{235}
679 \DeclareTextComposite{\'}{T1}{i}{236}
680 \DeclareTextComposite{\`}{T1}{i}{236}
681 \DeclareTextComposite{\'}{T1}{i}{237}
682 \DeclareTextComposite{\'}{T1}{i}{237}
683 \DeclareTextComposite{\^}{T1}{i}{238}
684 \DeclareTextComposite{\^}{T1}{i}{238}
685 \DeclareTextComposite{\\"}{T1}{i}{239}
686 \DeclareTextComposite{\\"}{T1}{i}{239}

"F0 = 240
687 \DeclareTextComposite{\~}{T1}{n}{241}
688 \DeclareTextComposite{\'}{T1}{o}{242}
689 \DeclareTextComposite{\'}{T1}{o}{243}
690 \DeclareTextComposite{\^}{T1}{o}{244}
691 \DeclareTextComposite{\~}{T1}{o}{245}
692 \DeclareTextComposite{\\"}{T1}{o}{246}

```

```

" F8 = 248

693 \DeclareTextComposite{\`}{T1}{u}{249}
694 \DeclareTextComposite{\'}{T1}{u}{250}
695 \DeclareTextComposite{\^}{T1}{u}{251}
696 \DeclareTextComposite{\"}{T1}{u}{252}
697 \DeclareTextComposite{\'}{T1}{y}{253}

698 \DeclareTextCompositeCommand{\k}{T1}{o}{\textogonekcentered{o}}
699 \DeclareTextCompositeCommand{\k}{T1}{O}{\textogonekcentered{O}}

700 \ifx\textcommaabove\@undefined\else
701 \DeclareTextCompositeCommand{\c}{T1}{g}{\textcommaabove{g}}
702 \fi
703 \ifx\textcommabelow\@undefined\else
704 \DeclareTextCompositeCommand{\c}{T1}{G}{\textcommabelow{G}}
705 \DeclareTextCompositeCommand{\c}{T1}{K}{\textcommabelow{K}}
706 \DeclareTextCompositeCommand{\c}{T1}{k}{\textcommabelow{k}}
707 \DeclareTextCompositeCommand{\c}{T1}{L}{\textcommabelow{L}}
708 \DeclareTextCompositeCommand{\c}{T1}{l}{\textcommabelow{l}}
709 \DeclareTextCompositeCommand{\c}{T1}{N}{\textcommabelow{N}}
710 \DeclareTextCompositeCommand{\c}{T1}{n}{\textcommabelow{n}}
711 \DeclareTextCompositeCommand{\c}{T1}{R}{\textcommabelow{R}}
712 \DeclareTextCompositeCommand{\c}{T1}{r}{\textcommabelow{r}}
713 \fi
714

```

## 1.7 Definitions for the OMS encoding

The definitions for the ‘ $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  math symbol’ (OMS) encoding. Even though this is meant to be a math font, it includes some of the standard  $\mathrm{L}\mathrm{A}\mathrm{T}_{\mathrm{E}}\mathrm{X}$  text symbols.

Declare the encoding.

```

715 {*OMS}
716 \DeclareFontEncoding{OMS}{}{}

```

Declare the symbols. Note that slot 13 has in places been named  $\backslash\mathrm{Orb}$ : please root out and destroy this impurity wherever you find it!

```

717 \DeclareTextSymbol{\textasteriskcentered}{OMS}{3} % "03
718 \DeclareTextSymbol{\textbackslash}{OMS}{110} % "6E
719 \DeclareTextSymbol{\textbar}{OMS}{106} % "6A
720 \DeclareTextSymbol{\textbardbl}{OMS}{107} % "6B
721 \DeclareTextSymbol{\textbraceleft}{OMS}{102} % "66
722 \DeclareTextSymbol{\textbraceright}{OMS}{103} % "67
723 \DeclareTextSymbol{\textbullet}{OMS}{15} % "0F
724 \DeclareTextSymbol{\textdaggerdbl}{OMS}{122} % "7A
725 \DeclareTextSymbol{\textdagger}{OMS}{121} % "79
726 \DeclareTextSymbol{\textparagraph}{OMS}{123} % "7B
727 \DeclareTextSymbol{\textperiodcentered}{OMS}{1} % "01
728 \DeclareTextSymbol{\textsection}{OMS}{120} % "78
729 \DeclareTextSymbol{\textbigcircle}{OMS}{13} % "0D
730 \DeclareTextCommand{\textcircled}{OMS}[1]{\hmode@bgroup
731 \ooalign{\%
732 \hfil\raise .07ex\hbox {\upshape#1}\hfil\crcr
733 \char 13 % "0D

```

```

734 }%
735 \egroup}
736
```

## 1.8 Definitions for the OML encoding

The definitions for the ‘ $\text{\TeX}$  math italic’ (OML) encoding. Even though this is meant to be a math font, it includes some of the standard  $\text{\LaTeX}$  text symbols.

Declare the encoding.

```

737 <*OML>
738 \DeclareFontEncoding{OML}{}{}

```

Declare the symbols.

```

739 \DeclareTextSymbol{\textless}{OML}{`<}
740 \DeclareTextSymbol{\textgreater}{OML}{`>}
741 \DeclareTextAccent{\t}{OML}{127} % "7F
742
```

## 1.9 Definitions for the OT4 encoding

These definitions are for the Polish extension to the ‘ $\text{\TeX}$  text’ (OT1) encoding. This encoding was created by B. Jackowski and M. Ryćko for use with the Polish version of Computer Modern and Computer Concrete. In positions 0–127 it is identical to OT1 but it contains some additional characters in the upper half. The  $\text{\LaTeX}$  support was developed by Mariusz Olko.

The PL fonts that use it are available as follows:

Metafont sources <ftp://ftp.gust.org.pl/TeX/language/polish/pl-mf.zip>;

Font files <ftp://ftp.gust.org.pl/TeX/language/polish/pl-tfm.zip>.

Declare the encoding.

```

743 <*OT4>
744 \DeclareFontEncoding{OT4}{}{}
745 \DeclareFontSubstitution{OT4}{cmr}{m}{n}

```

Declare the accents.

```

746 \DeclareTextAccent{\"}{OT4}{127}
747 \DeclareTextAccent{\'}{OT4}{19}
748 \DeclareTextAccent{\.}{OT4}{95}
749 \DeclareTextAccent{\=}{OT4}{22}
750 \DeclareTextAccent{\^}{OT4}{94}
751 \DeclareTextAccent{\`}{OT4}{18}
752 \DeclareTextAccent{\~}{OT4}{126}
753 \DeclareTextAccent{\H}{OT4}{125}
754 \DeclareTextAccent{\u}{OT4}{21}
755 \DeclareTextAccent{\v}{OT4}{20}
756 \DeclareTextAccent{\r}{OT4}{23}

```

The ogonek accent is available only under a e A & E. But we have to provide some definition for \k. Some other accents have to be built by hand as in OT1:

```

757 \DeclareTextCommand{\k}{OT4}[1]{%
758 \TextSymbolUnavailable{\k{#1}}#1}

```

In these definitions we no longer use the helper function `\sh@ft` from plain.tex since that now has two incompatible definitions.

```

759 \DeclareTextCommand{\b}{OT4}[1]
760 {\hmode@bgroup\o@align{\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@align{\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{\texttwelvedash}{TS1}{21}
858 \DeclareTextSymbol{\textthreequartersemdash}{TS1}{22}
"18 = 24
859 \DeclareTextSymbol{\textleftarrow}{TS1}{24}
860 \DeclareTextSymbol{\textrightarrow}{TS1}{25}
"20 = 32
861 \DeclareTextSymbol{\textblank}{TS1}{32}
862 \DeclareTextSymbol{\textdollar}{TS1}{36}
863 \DeclareTextSymbol{\textquotesingle}{TS1}{39}
"28 = 40

```

The symbol `\textasteriskcentered` “\*” is supposed to be always available in `TS1` and that is important as it is used in footnote symbols. However, in a few fonts it is missing even though they are otherwise fairly complete. We therefore use a rather elaborate method and check if the slot has a glyph and if not produce a poor man’s version by using a normal “\*” slightly enlarged and somewhat lowered. The main application for this symbol is in footnote symbols and there it should produce a comparable size and show a similar placement.

```

864 \% \DeclareTextSymbol{\textasteriskcentered}{TS1}{42} % that's wanted
865 \DeclareTextCommand \textasteriskcentered{TS1}{% % and that's needed
866 \iffontchar\font 42 \char42 \else
867 \begingroup\fontencoding{T1}%
868 \fontsize

```

```

869 {\the\dimexpr1.3\dimexpr\f@size pt\relax}%
870 {\f@baselineskip}%
871 \selectfont
872 \raisebox{-0.7ex}{[\dimexpr\height-0.7ex][0pt]{*}}%
873 \endgroup
874 \fi
875 }

Note that '054 is a comma and '056 is a full stop: these make numbers using oldstyle digits easier to input.

876 \DeclareTextSymbol{\textdblhyphen}{TS1}{45}
877 \DeclareTextSymbol{\textfractionsolidus}{TS1}{47}

Oldstyle digits.

'30 = 48

878 \DeclareTextSymbol{\textzerooldstyle}{TS1}{48}
879 \DeclareTextSymbol{\textoneoldstyle}{TS1}{49}
880 \DeclareTextSymbol{\texttwooldstyle}{TS1}{50}
881 \DeclareTextSymbol{\textthreeoldstyle}{TS1}{51}
882 \DeclareTextSymbol{\textfouroldstyle}{TS1}{52}
883 \DeclareTextSymbol{\textfiveoldstyle}{TS1}{53}
884 \DeclareTextSymbol{\textsixoldstyle}{TS1}{54}
885 \DeclareTextSymbol{\textsevenoldstyle}{TS1}{55}

'38 = 56

886 \DeclareTextSymbol{\texteightoldstyle}{TS1}{56}
887 \DeclareTextSymbol{\textnineoldstyle}{TS1}{57}

More text companion symbols.

888 \DeclareTextSymbol{\textlangel}{TS1}{60}
889 \DeclareTextSymbol{\textminus}{TS1}{61}
890 \DeclareTextSymbol{\textrangle}{TS1}{62}

'48 = 72

891 \DeclareTextSymbol{\textmho}{TS1}{77}

The big circle is here to define the command \textcircled. Formerly it was taken from the cmsy font.

892 \DeclareTextSymbol{\textbigcircle}{TS1}{79}
893 \DeclareTextCommand{\textcircled}{TS1}[1]{\hmode@bgroup
894 \oalign{%
895 \hfil \raise .07ex\hbox {\upshape#1}\hfil \crcr
896 \char 79 % '117 = "4F
897 }%
898 \egroup}

More text companion symbols.

'50 = 80

899 \DeclareTextSymbol{\textohm}{TS1}{87}

'58 = 88

900 \DeclareTextSymbol{\textlbrackdbl}{TS1}{91}
901 \DeclareTextSymbol{\textrbrackdbl}{TS1}{93}
902 \DeclareTextSymbol{\textuparrowarrow}{TS1}{94}
903 \DeclareTextSymbol{\textdownarrowarrow}{TS1}{95}

```

```

"60 = 96
904 \DeclareTextSymbol{\textasciigrave}{TS1}{96}
905 \DeclareTextSymbol{\textborn}{TS1}{98}
906 \DeclareTextSymbol{\textdivorced}{TS1}{99}
907 \DeclareTextSymbol{\textdied}{TS1}{100}

"68 = 104
908 \DeclareTextSymbol{\textleaf}{TS1}{108}
909 \DeclareTextSymbol{\textmarried}{TS1}{109}
910 \DeclareTextSymbol{\textmusicalnote}{TS1}{110}

"78 = 120
911 \DeclareTextSymbol{\texttildelow}{TS1}{126}
This glyph, \textdblhyphenchar is hanging, like the hyphenchar of the ec fonts.
912 \DeclareTextSymbol{\textdblhyphenchar}{TS1}{127}

"80 = 128
913 \DeclareTextSymbol{\textasciibreve}{TS1}{128}
914 \DeclareTextSymbol{\textasciicaron}{TS1}{129}
This next glyph is not the same as \textquotedbl.
915 \DeclareTextSymbol{\textacutedbl}{TS1}{130}
916 \DeclareTextSymbol{\textgravedbl}{TS1}{131}
917 \DeclareTextSymbol{\textdagger}{TS1}{132}
918 \DeclareTextSymbol{\textdaggerdbl}{TS1}{133}
919 \DeclareTextSymbol{\textbardbl}{TS1}{134}
920 \DeclareTextSymbol{\textperthousand}{TS1}{135}

"88 = 136
921 \DeclareTextSymbol{\textbullet}{TS1}{136}
922 \DeclareTextSymbol{\textcelsius}{TS1}{137}
923 \DeclareTextSymbol{\textdollaroldstyle}{TS1}{138}
924 \DeclareTextSymbol{\textcentoldstyle}{TS1}{139}
925 \DeclareTextSymbol{\textflorin}{TS1}{140}
926 \DeclareTextSymbol{\textcolonmonetary}{TS1}{141}
927 \DeclareTextSymbol{\textwon}{TS1}{142}
928 \DeclareTextSymbol{\textnaira}{TS1}{143}

"90 = 144
929 \DeclareTextSymbol{\textguarani}{TS1}{144}
930 \DeclareTextSymbol{\textpeso}{TS1}{145}
931 \DeclareTextSymbol{\textlira}{TS1}{146}
932 \DeclareTextSymbol{\textrecipe}{TS1}{147}
933 \DeclareTextSymbol{\textinterrobang}{TS1}{148}
934 \DeclareTextSymbol{\textinterrobangdown}{TS1}{149}
935 \DeclareTextSymbol{\textdong}{TS1}{150}
936 \DeclareTextSymbol{\texttrademark}{TS1}{151}

"98 = 152
937 \DeclareTextSymbol{\textpertenthousand}{TS1}{152}
938 \DeclareTextSymbol{\textpilcrow}{TS1}{153}
939 \DeclareTextSymbol{\textbaht}{TS1}{154}
940 \DeclareTextSymbol{\textnumero}{TS1}{155}

```

This next name may change. For the following sign we know only a german name, which is abziglich. The meaning is something like “commercial minus”. An ASCII ersatz is ./ (dot slash dot). The temporary English name is \textdiscount.

```

941 \DeclareTextSymbol{\textdiscount}{TS1}{156}
942 \DeclareTextSymbol{\textestimated}{TS1}{157}
943 \DeclareTextSymbol{\textopenbullet}{TS1}{158}
944 \DeclareTextSymbol{\textservicemark}{TS1}{159}

"A0 = 160
945 \DeclareTextSymbol{\textlquill}{TS1}{160}
946 \DeclareTextSymbol{\textrquill}{TS1}{161}
947 \DeclareTextSymbol{\textcent}{TS1}{162}
948 \DeclareTextSymbol{\textsterling}{TS1}{163}
949 \DeclareTextSymbol{\textcurrency}{TS1}{164}
950 \DeclareTextSymbol{\textyen}{TS1}{165}
951 \DeclareTextSymbol{\textbrokenbar}{TS1}{166}
952 \DeclareTextSymbol{\textsection}{TS1}{167}

"A8 = 168
953 \DeclareTextSymbol{\textasciidieresis}{TS1}{168}
954 \DeclareTextSymbol{\textcopyright}{TS1}{169}
955 \DeclareTextSymbol{\textordfeminine}{TS1}{170}
956 \DeclareTextSymbol{\textcopyleft}{TS1}{171}
957 \DeclareTextSymbol{\textlnot}{TS1}{172}

The meaning of the circled-P is “sound recording copyright”.

958 \DeclareTextSymbol{\textcircledP}{TS1}{173}
959 \DeclareTextSymbol{\textregistered}{TS1}{174}
960 \DeclareTextSymbol{\textasciimacron}{TS1}{175}

"B0 = 176
961 \DeclareTextSymbol{\textdegree}{TS1}{176}
962 \DeclareTextSymbol{\textpm}{TS1}{177}
963 \DeclareTextSymbol{\texttwosuperior}{TS1}{178}
964 \DeclareTextSymbol{\textthreesuperior}{TS1}{179}
965 \DeclareTextSymbol{\textasciacute}{TS1}{180}
966 \DeclareTextSymbol{\textmu}{TS1}{181} % micro sign
967 \DeclareTextSymbol{\textparagraph}{TS1}{182}
968 \DeclareTextSymbol{\textperiodcentered}{TS1}{183}

"B8 = 184
969 \DeclareTextSymbol{\textreferencemark}{TS1}{184}
970 \DeclareTextSymbol{\textonesuperior}{TS1}{185}
971 \DeclareTextSymbol{\textordmasculine}{TS1}{186}
972 \DeclareTextSymbol{\textsurd}{TS1}{187}
973 \DeclareTextSymbol{\textonequarter}{TS1}{188}
974 \DeclareTextSymbol{\textonehalf}{TS1}{189}
975 \DeclareTextSymbol{\textthreequarters}{TS1}{190}
976 \DeclareTextSymbol{\texteuro}{TS1}{191}

"E0 = 208
977 \DeclareTextSymbol{\texttimes}{TS1}{214}

"F0 = 240
978 \DeclareTextSymbol{\textdiv}{TS1}{246}
979 </TS1>

```

## 1.11 Definitions for the TU encoding

The TU encoding was originally introduced in the contributed package `fontspec` as a Unicode encoding for XeTeX and LuaTeX.

Normally for these engines, the input consists of Unicode characters encoded in UTF-8. There is therefore little need to use the traditional (ASCII) encoding-specific commands

However, sometimes (e.g. for backwards compatibility) it can be useful to access these Unicode characters via such ASCII-based markup. The commands provided here cover the characters in the T1 and TS1 encodings, but specified in Unicode position. Almost all the command names have been mechanically extracted from the `inputenc` UTF-8 support, which is essentially doing a reverse mapping from UTF-8 data to L<sup>A</sup>T<sub>E</sub>X L<sup>I</sup>C<sup>R</sup> 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;}
994 \ifnum\luatexversion<110
995 \def\reserved@a{\%
996 \def\@remove@tlig##1{\@remove@tlig##1\@nil\@nil\relax}
997 \def\@remove@tlig##1#1{\@remove@tlig##1#1}
998 \edef\reserved@b{\detokenize{+tlig;}}
999 \expandafter\reserved@a\expandafter{\reserved@b}
1000 \def\@remove@tlig##1\@nil#2\relax{#1}
```

```

1001 \def\remove@tlig#1{%
1002 \begingroup
1003 \font\remove@tlig
1004 \expandafter\@remove@tlig\expandafter{\fontname\font}%
1005 \remove@tlig
1006 \char#1\relax
1007 \endgroup
1008 }
1009 \else
1010 \newprotectedluacmd\@remove@tlig@@@%

```

Now we can define the function. Mostly we just have to insert a protected glyph node, which is a glyph node with subtype 256. But we have to keep track of the current mode to avoid inserting the glyph into a vlist.

```

1011 \now@and@everyjob{\directlua{
1012 local rawchar_func = token.create'@remove@tlig@@@'.index
1013 local forcehmode = tex.forcehmode
1014 local put_next = token.put_next
1015 local glyph_id = node.id'glyph'
1016 local rawchar_token = token.new(rawchar_func, token.command_id'lua_call')
1017 lua.get_functions_table()[rawchar_func] = function()
1018 local mode = tex.nest.top.mode
1019 if mode == 1 or mode == -1 then
1020 put_next(rawchar_token)
1021 return forcehmode(true)
1022 end
1023 local n = node.new(glyph_id, 256)
1024 n.font = font.current()
1025 n.char = token.scan_int()
1026 return node.write(n)
1027 end
1028 }}

```

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

```

1029 \def\remove@tlig#1{\@remove@tlig@@@#1\relax}
1030 \fi
1031 \fi
1032 \else
XeTeX
1033 \def\UnicodeFontTeXLigatures{mapping=tex-text;}
1034 \def\remove@tlig#1{\XeTeXglyph\numexpr\XeTeXcharglyph#1\relax}
1035 \fi
1036 \def\UnicodeFontFile#1#2{"[#1]:#2"}
1037 \def\UnicodeFontName#1#2{"#1:#2"}

 Declare the encoding
1038 \DeclareFontEncoding\UnicodeEncodingName{}{}

 Declare accent command to use a postspended combining character rather than the
TeX \accent primitive
1039 \def\add@unicode@accent#1#2{%
1040 \if\relax\detokenize{#2}\relax^\wedge a0\else#2\fi
1041 \char#1\relax}

```

In its original implementation `\DeclareUnicodeAccent` was given 3 arguments (with second the “Unicode encoding” a.k.a., `\UnicodeEncodingName`) while in other places, e.g., `\DeclareUnicodeComposite`, we always made encoding implicit. So we now change it here to implicit too so that the interfaces become a bit more consistent. To avoid making that a breaking change (even though it only affects two packages on CTAN) we test for #2 being `\UnicodeEncodingName`. This would not catch if somebody used `\DeclareUnicodeAccent{\=}{TU-sub}{“0304”}` but that fortunately hasn’t happened. With the implicit argument you would need to change `\UnicodeEncodingName` instead, as you have to do anyway for the other interface commands.

```

1042 \def\DeclareUnicodeAccent#1#2{%
1043 \edef\reserved@a{#2}%
1044 \edef\reserved@b{\UnicodeEncodingName}%
1045 \ifx\reserved@a\reserved@b
1046 \def\reserved@a{\DeclareUnicodeAccent@{#1}}%
1047 \else
1048 \def\reserved@a{\DeclareUnicodeAccent@{#1}\UnicodeEncodingName}%
1049 \fi
1050 \reserved@a{#2}%
1051 }
1052 \def\DeclareUnicodeAccent@#1#2#3{%
1053 \DeclareTextCommand{#1}{#2}{\addunicode@accent{#3}}%
1054 }
```

Wrapper around `\DeclareTextCompositeCommand` that uses the declared composite if it exists in the current font or falls back to the default definition for the TU accent if not.

```

1055 {
1056 \catcode\z@=11\relax
1057 \gdef\DeclareUnicodeComposite#1#2#3{%
1058 \def\reserved@a##1##2{%
1059 \DeclareTextCompositeCommand#1\UnicodeEncodingName{#2}{%
1060 \iffontchar\font#3 ##2%
1061 \else ##1\fi}%
1062 \expandafter\expandafter\expandafter\extract@default@composite
1063 \csname\UnicodeEncodingName\string#1\endcsname{#2}\@nil
1064 \bgroup
1065 \lccode\z@#3 %
1066 \lowercase{\egroup
1067 \expandafter\reserved@a\expandafter{\reserved@b}{^{\z@}}}}%
1068 }
1069 \def\extract@default@composite#1{%
1070 \ifx\@text@composite#1%
1071 \expandafter\extract@default@composite@a
1072 \else
1073 \expandafter\extract@default@composite@b\expandafter#1%
1074 \fi}
1075 \def\extract@default@composite@a#1\@text@composite#2\@nil{%
1076 \def\reserved@b{#2}}
1077 \def\extract@default@composite@b#1#2\@nil{%
1078 \def\reserved@b{#1#2}}
```

Next two commands are simply syntactic sugar to go with the other `\DeclareUnicode...` declarations.

```

1079 \def\DeclareUnicodeSymbol#1{\DeclareTextSymbol{#1}{\UnicodeEncodingName}}
1080 \def\DeclareUnicodeCommand#1{\DeclareTextCommand{#1}{\UnicodeEncodingName}}
1081 \DeclareUnicodeCommand{textquotesingle} {\remove@tlig{"0027}}
1082 \DeclareUnicodeCommand{textasciigrave} {\remove@tlig{"0060}}
1083 \DeclareUnicodeCommand{textquotedbl} {\remove@tlig{"0022}}
1084 \DeclareUnicodeSymbol{textdollar} {"0024}
1085 \DeclareUnicodeSymbol{textless} {"003C}
1086 \DeclareUnicodeSymbol{textgreater} {"003E}
1087 \DeclareUnicodeSymbol{textbackslash} {"005C}
1088 \DeclareUnicodeSymbol{textasciicircum} {"005E}
1089 \DeclareUnicodeSymbol{textunderscore} {"005F}
1090 \DeclareUnicodeSymbol{textbraceleft} {"007B}
1091 \DeclareUnicodeSymbol{textbar} {"007C}
1092 \DeclareUnicodeSymbol{textbraceright} {"007D}
1093 \DeclareUnicodeSymbol{textasciitilde} {"007E}
1094 \DeclareUnicodeSymbol{textexclamdown} {"00A1}
1095 \DeclareUnicodeSymbol{textcent} {"00A2}
1096 \DeclareUnicodeSymbol{textsterling} {"00A3}
1097 \DeclareUnicodeSymbol{textcurrency} {"00A4}
1098 \DeclareUnicodeSymbol{textyen} {"00A5}
1099 \DeclareUnicodeSymbol{textbrokenbar} {"00A6}
1100 \DeclareUnicodeSymbol{textsection} {"00A7}
1101 \DeclareUnicodeSymbol{textasciidieresis} {"00A8}
1102 \DeclareUnicodeSymbol{textcopyright} {"00A9}
1103 \DeclareUnicodeSymbol{textordfeminine} {"00AA}

1104 \DeclareUnicodeSymbol{guillemetleft} {"00AB}
1105 % old Adobe name
1106 \DeclareUnicodeSymbol{guillemotleft} {"00AB}
1107 \DeclareUnicodeSymbol{textlnot} {"00AC}
1108 \DeclareUnicodeSymbol{textregistered} {"00AE}
1109 \DeclareUnicodeSymbol{textasciimacron} {"00AF}
1110 \DeclareUnicodeSymbol{textdegree} {"00B0}
1111 \DeclareUnicodeSymbol{textpm} {"00B1}
1112 \DeclareUnicodeSymbol{texttwosuperior} {"00B2}
1113 \DeclareUnicodeSymbol{textthreesuperior} {"00B3}
1114 \DeclareUnicodeSymbol{textasciacute} {"00B4}
1115 \DeclareUnicodeSymbol{textmu} {"00B5}
1116 \DeclareUnicodeSymbol{textparagraph} {"00B6}
1117 \DeclareUnicodeSymbol{textperiodcentered} {"00B7}
1118 \DeclareUnicodeSymbol{textonesuperior} {"00B9}
1119 \DeclareUnicodeSymbol{textordmasculine} {"00BA}

1120 \DeclareUnicodeSymbol{guillemetright} {"00BB}
1121 % old Adobe name
1122 \DeclareUnicodeSymbol{guillemotright} {"00BB}
1123 \DeclareUnicodeSymbol{textonequarter} {"00BC}
1124 \DeclareUnicodeSymbol{textonehalf} {"00BD}
1125 \DeclareUnicodeSymbol{textthreequarters} {"00BE}
1126 \DeclareUnicodeSymbol{textquestiondown} {"00BF}
1127 \DeclareUnicodeSymbol{AE} {"00C6}
1128 \DeclareUnicodeSymbol{DH} {"00D0}
1129 \DeclareUnicodeSymbol{texttimes} {"00D7}

```

```

1130 \DeclareUnicodeSymbol{\O} {"00D8}
1131 \DeclareUnicodeSymbol{\TH} {"00DE}
1132 \DeclareUnicodeSymbol{\ss} {"00DF}
1133 \DeclareUnicodeSymbol{\ae} {"00E6}
1134 \DeclareUnicodeSymbol{\dh} {"00F0}
1135 \DeclareUnicodeSymbol{\textdiv} {"00F7}
1136 \DeclareUnicodeSymbol{\o} {"00F8}
1137 \DeclareUnicodeSymbol{\th} {"00FE}
1138 \DeclareUnicodeSymbol{\DJ} {"0110}
1139 \DeclareUnicodeSymbol{\dj} {"0111}
1140 \DeclareUnicodeSymbol{\i} {"0131}
1141 \DeclareUnicodeSymbol{\IJ} {"0132}
1142 \DeclareUnicodeSymbol{\ij} {"0133}
1143 \DeclareUnicodeSymbol{\L} {"0141}
1144 \DeclareUnicodeSymbol{\l} {"0142}
1145 \DeclareUnicodeSymbol{\NG} {"014A}
1146 \DeclareUnicodeSymbol{\ng} {"014B}
1147 \DeclareUnicodeSymbol{\OE} {"0152}
1148 \DeclareUnicodeSymbol{\oe} {"0153}
1149 \DeclareUnicodeSymbol{\textflorin} {"0192}
1150 \DeclareUnicodeSymbol{\j} {"0237}
1151 \DeclareUnicodeSymbol{\textasciicaron} {"02C7}
1152 \DeclareUnicodeSymbol{\textasciibreve} {"02D8}
1153 \DeclareUnicodeSymbol{\textacutedbl} {"02DD}
1154 \DeclareUnicodeSymbol{\textgravedbl} {"02F5}
1155 \DeclareUnicodeSymbol{\texttildelow} {"02F7}
1156 \DeclareUnicodeSymbol{\textbaht} {"0E3F}
1157 \DeclareUnicodeSymbol{\SS} {"1E9E}
1158 \DeclareUnicodeSymbol{\textcompwordmark} {"200C}

1159 %\DeclareUnicodeSymbol{\textnonbreakinghyphen} {"2011}
1160 %\DeclareUnicodeSymbol{\textfiguredash} {"2012}
1161 \DeclareUnicodeSymbol{\textendash} {"2013}
1162 \DeclareUnicodeSymbol{\textemdash} {"2014}
1163 %\DeclareUnicodeSymbol{\texthorizontalbar} {"2015}

```

Unfortunately some fonts do not implement "2011, "2012 and/or "2015 (including the L<sup>A</sup>T<sub>E</sub>X default fonts for Unicode engines) so we provide some approximations if the glyph is missing, like we do for OT1 and T1.

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

```

1164 \DeclareUnicodeCommand{\textnonbreakinghyphen}
1165 {\iffontchar\font "2011 \char "2011 \else \mbox{-}\nobreak\hspace{z@} \fi}
1166 \DeclareUnicodeCommand{\textfiguredash}
1167 {\iffontchar\font "2012 \char "2012 \else \char "2013 \fi}
1168 \DeclareUnicodeCommand{\texthorizontalbar}
1169 {\iffontchar\font "2015 \char "2015 \else \char "2014 \fi}

1170 \DeclareUnicodeSymbol{\textbardbl} {"2016}
1171 \DeclareUnicodeSymbol{\textquotel} {"2018}
1172 \DeclareUnicodeSymbol{\textquoter} {"2019}
1173 \DeclareUnicodeSymbol{\quotesinglbase} {"201A}
1174 \DeclareUnicodeSymbol{\textquotedblleft} {"201C}
1175 \DeclareUnicodeSymbol{\textquotedblright} {"201D}
1176 \DeclareUnicodeSymbol{\quotedblbase} {"201E}

```

```

1177 \DeclareUnicodeSymbol{\textdagger} {"2020}
1178 \DeclareUnicodeSymbol{\textdaggerdbl} {"2021}
1179 \DeclareUnicodeSymbol{\textbullet} {"2022}
1180 \DeclareUnicodeSymbol{\textellipsis} {"2026}
1181 \DeclareUnicodeSymbol{\textperthousand} {"2030}
1182 \DeclareUnicodeSymbol{\textpertenthousand} {"2031}
1183 \DeclareUnicodeSymbol{\guilsinglleft} {"2039}
1184 \DeclareUnicodeSymbol{\guilsinglright} {"203A}
1185 \DeclareUnicodeSymbol{\textreferencemark} {"203B}
1186 \DeclareUnicodeSymbol{\textinterrobang} {"203D}
1187 \DeclareUnicodeSymbol{\textfractionsolidus} {"2044}
1188 \DeclareUnicodeSymbol{\textlquill} {"2045}
1189 \DeclareUnicodeSymbol{\textrquill} {"2046}
1190 \DeclareUnicodeSymbol{\textdiscount} {"2052}
1191 \DeclareUnicodeSymbol{\textcolonmonetary} {"20A1}
1192 \DeclareUnicodeSymbol{\textlira} {"20A4}
1193 \DeclareUnicodeSymbol{\textnaira} {"20A6}
1194 \DeclareUnicodeSymbol{\textwon} {"20A9}
1195 \DeclareUnicodeSymbol{\textdong} {"20AB}
1196 \DeclareUnicodeSymbol{\texteuro} {"20AC}
1197 \DeclareUnicodeSymbol{\textpeso} {"20B1}
1198 \DeclareUnicodeSymbol{\textcelsius} {"2103}
1199 \DeclareUnicodeSymbol{\textnumero} {"2116}
1200 \DeclareUnicodeSymbol{\textcircledP} {"2117}
1201 \DeclareUnicodeSymbol{\textrecipie} {"211E}
1202 \DeclareUnicodeSymbol{\textservicemark} {"2120}
1203 \DeclareUnicodeSymbol{\texttrademark} {"2122}
1204 \DeclareUnicodeSymbol{\textohm} {"2126}
1205 \DeclareUnicodeSymbol{\textmho} {"2127}
1206 \DeclareUnicodeSymbol{\textestimated} {"212E}
1207 \DeclareUnicodeSymbol{\textleftarrow} {"2190}
1208 \DeclareUnicodeSymbol{\textuparrow} {"2191}
1209 \DeclareUnicodeSymbol{\textrightarrow} {"2192}
1210 \DeclareUnicodeSymbol{\textdownarrow} {"2193}
1211 \DeclareUnicodeSymbol{\textminus} {"2212}
1212
1213 \DeclareUnicodeSymbol{\Hwithstroke} {"0126}
1214 \DeclareUnicodeSymbol{\hwithstroke} {"0127}

```

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

```

1215 \DeclareUnicodeCommand{\textasteriskcentered}{%
1216 \iffontchar\font"2217 \char"2217 \else
1217 \begingroup
1218 \fontsize
1219 {\the\dimexpr1.3\dimexpr\f@size pt\relax}%
1220 {\f@baselineskip}%
1221 \selectfont
1222 \raisebox{-0.7ex}{[\dimexpr\height-0.7ex] [0pt]{*}}%
1223 \endgroup
1224 \fi
1225 }
1226 \DeclareUnicodeSymbol{\textsurd} {"221A}
1227 \DeclareUnicodeSymbol{\textlangle} {"2329}

```

```

1228 \DeclareUnicodeSymbol{\textrangle} {"232A}
1229 \DeclareUnicodeSymbol{\textblank} {"2422}
1230 \DeclareUnicodeSymbol{\textvisiblespace} {"2423}
1231 \DeclareUnicodeSymbol{\textopenbullet} {"25E6}
1232 \DeclareUnicodeSymbol{\textbigcircle} {"25EF}
1233 \DeclareUnicodeSymbol{\textmusicalnote} {"266A}
1234 \DeclareUnicodeSymbol{\textmarried} {"26AD}
1235 \DeclareUnicodeSymbol{\textdivorced} {"26AE}
1236 \DeclareUnicodeSymbol{\textinterrobangdown} {"2E18}

```

Accents must be declared before the composites that use them.

```

1237 \DeclareUnicodeAccent{\`}{0300}
1238 \DeclareUnicodeAccent{\'}{0301}
1239 \DeclareUnicodeAccent{\^}{0302}
1240 \DeclareUnicodeAccent{\~}{0303}
1241 \DeclareUnicodeAccent{\=}{0304}
1242 \DeclareUnicodeAccent{\u}{0306}
1243 \DeclareUnicodeAccent{\.}{0307}
1244 \DeclareUnicodeAccent{\"}{0308}
1245 \DeclareUnicodeAccent{\r}{030A}
1246 \DeclareUnicodeAccent{\H}{030B}
1247 \DeclareUnicodeAccent{\v}{030C}
1248 \DeclareUnicodeAccent{\b}{0332}
1249 \DeclareUnicodeAccent{\d}{0323}
1250 \DeclareUnicodeAccent{\c}{0327}
1251 \DeclareUnicodeAccent{\k}{0328}

```

The odd one out:

```

1252 \DeclareUnicodeCommand{textcommabelow[1]
1253 {\hmode@bgroup\oalign{\null#1\crcr\hidewidth\raise-.31ex
1254 \hbox{\check@mathfonts\fontsize\ssf@size\z@
1255 \math@fontsfalse\selectfont,\}\hidewidth}\egroup}
1256 \DeclareUnicodeComposite{\^} {"005E}
1257 \DeclareUnicodeComposite{\~} {"007E}
1258 \DeclareUnicodeComposite{\`} {A>{"00C0}
1259 \DeclareUnicodeComposite{\'} {A>{"00C1}
1260 \DeclareUnicodeComposite{\^} {A>{"00C2}
1261 \DeclareUnicodeComposite{\~} {A>{"00C3}
1262 \DeclareUnicodeComposite{\"} {A>{"00C4}
1263 \DeclareUnicodeComposite{\r} {A>{"00C5}
1264 \DeclareUnicodeComposite{\c} {C>{"00C7}
1265 \DeclareUnicodeComposite{\`} {E>{"00C8}
1266 \DeclareUnicodeComposite{\'} {E>{"00C9}
1267 \DeclareUnicodeComposite{\^} {E>{"00CA}
1268 \DeclareUnicodeComposite{\"} {E>{"00CB}
1269 \DeclareUnicodeComposite{\`} {I>{"00CC}
1270 \DeclareUnicodeComposite{\'} {I>{"00CD}
1271 \DeclareUnicodeComposite{\~} {I>{"00CE}
1272 \DeclareUnicodeComposite{\"} {I>{"00CF}
1273 \DeclareUnicodeComposite{\~} {N>{"00D1}
1274 \DeclareUnicodeComposite{\`} {O>{"00D2}
1275 \DeclareUnicodeComposite{\'} {O>{"00D3}
1276 \DeclareUnicodeComposite{\^} {O>{"00D4}
1277 \DeclareUnicodeComposite{\~} {O>{"00D5}

```

```

1278 \DeclareUnicodeComposite{\"}
1279 \DeclareUnicodeComposite{\'}
1280 \DeclareUnicodeComposite{\`}
1281 \DeclareUnicodeComposite{\`}
1282 \DeclareUnicodeComposite{\"}
1283 \DeclareUnicodeComposite{\`}
1284 \DeclareUnicodeComposite{\'}
1285 \DeclareUnicodeComposite{\`}
1286 \DeclareUnicodeComposite{\`}
1287 \DeclareUnicodeComposite{\`}
1288 \DeclareUnicodeComposite{\"}
1289 \DeclareUnicodeComposite{\r}
1290 \DeclareUnicodeComposite{\c}
1291 \DeclareUnicodeComposite{\`}
1292 \DeclareUnicodeComposite{\`}
1293 \DeclareUnicodeComposite{\`}
1294 \DeclareUnicodeComposite{\"}
1295 \DeclareUnicodeComposite{\'}
1296 \DeclareUnicodeComposite{\'}
1297 \DeclareUnicodeComposite{\`}
1298 \DeclareUnicodeComposite{\`}
1299 \DeclareUnicodeComposite{\`}
1300 \DeclareUnicodeComposite{\`}
1301 \DeclareUnicodeComposite{\"}
1302 \DeclareUnicodeComposite{\"}
1303 \DeclareUnicodeComposite{\`}
1304 \DeclareUnicodeComposite{\`}
1305 \DeclareUnicodeComposite{\`}
1306 \DeclareUnicodeComposite{\`}
1307 \DeclareUnicodeComposite{\`}
1308 \DeclareUnicodeComposite{\"}
1309 \DeclareUnicodeComposite{\`}
1310 \DeclareUnicodeComposite{\`}
1311 \DeclareUnicodeComposite{\`}
1312 \DeclareUnicodeComposite{\"}
1313 \DeclareUnicodeComposite{\`}
1314 \DeclareUnicodeComposite{\"}
1315 \DeclareUnicodeComposite{\=}
1316 \DeclareUnicodeComposite{\=}
1317 \DeclareUnicodeComposite{\u}
1318 \DeclareUnicodeComposite{\u}
1319 \DeclareUnicodeComposite{\k}
1320 \DeclareUnicodeComposite{\k}
1321 \DeclareUnicodeComposite{\`}
1322 \DeclareUnicodeComposite{\`}
1323 \DeclareUnicodeComposite{\`}
1324 \DeclareUnicodeComposite{\`}
1325 \DeclareUnicodeComposite{\.}
1326 \DeclareUnicodeComposite{\.}
1327 \DeclareUnicodeComposite{\v}
1328 \DeclareUnicodeComposite{\v}
1329 \DeclareUnicodeComposite{\v}
1330 \DeclareUnicodeComposite{\v}
1331 \DeclareUnicodeComposite{\=}
{O}{"00D6}
{U}{"00D9}
{U}{"00DA}
{U}{"00DB}
{U}{"00DC}
{Y}{"00DD}
{a}{"00E0}
{a}{"00E1}
{a}{"00E2}
{a}{"00E3}
{a}{"00E4}
{a}{"00E5}
{c}{"00E7}
{e}{"00E8}
{e}{"00E9}
{e}{"00EA}
{e}{"00EB}
\i {"00EC}
{i}{"00EC}
\i {"00ED}
{i}{"00ED}
\i {"00EE}
{i}{"00EE}
\i {"00EF}
{i}{"00EF}
{n}{"00F1}
{o}{"00F2}
{o}{"00F3}
{o}{"00F4}
{o}{"00F5}
{o}{"00F6}
{u}{"00F9}
{u}{"00FA}
{u}{"00FB}
{u}{"00FC}
{y}{"00FD}
{y}{"00FF}
{A}{"0100}
{a}{"0101}
{A}{"0102}
{a}{"0103}
{A}{"0104}
{a}{"0105}
{C}{"0106}
{c}{"0107}
{C}{"0108}
{c}{"0109}
{C}{"010A}
{c}{"010B}
{C}{"010C}
{c}{"010D}
{D}{"010E}
{d}{"010F}
{E}{"0112}

```

```

1332 \DeclareUnicodeComposite{\=}{e}{0113}
1333 \DeclareUnicodeComposite{\u}{E}{0114}
1334 \DeclareUnicodeComposite{\u}{e}{0115}
1335 \DeclareUnicodeComposite{\.}{E}{0116}
1336 \DeclareUnicodeComposite{\.}{e}{0117}
1337 \DeclareUnicodeComposite{\k}{E}{0118}
1338 \DeclareUnicodeComposite{\k}{e}{0119}
1339 \DeclareUnicodeComposite{\v}{E}{011A}
1340 \DeclareUnicodeComposite{\v}{e}{011B}
1341 \DeclareUnicodeComposite{\^}{G}{011C}
1342 \DeclareUnicodeComposite{\^}{g}{011D}
1343 \DeclareUnicodeComposite{\u}{G}{011E}
1344 \DeclareUnicodeComposite{\u}{g}{011F}
1345 \DeclareUnicodeComposite{\.}{G}{0120}
1346 \DeclareUnicodeComposite{\.}{g}{0121}
1347 \DeclareUnicodeComposite{\c}{G}{0122}
1348 \DeclareUnicodeComposite{\c}{g}{0123}
1349 \DeclareUnicodeComposite{\^}{H}{0124}
1350 \DeclareUnicodeComposite{\^}{h}{0125}
1351 \DeclareUnicodeComposite{\~}{I}{0128}
1352 \DeclareUnicodeComposite{\~}{\i}{0129}
1353 \DeclareUnicodeComposite{\~}{i}{0129}
1354 \DeclareUnicodeComposite{\=}{I}{012A}
1355 \DeclareUnicodeComposite{\=}{\i}{012B}
1356 \DeclareUnicodeComposite{\=}{i}{012B}
1357 \DeclareUnicodeComposite{\u}{I}{012C}
1358 \DeclareUnicodeComposite{\u}{\i}{012D}
1359 \DeclareUnicodeComposite{\u}{i}{012D}
1360 \DeclareUnicodeComposite{\k}{I}{012E}
1361 \DeclareUnicodeComposite{\k}{\i}{012F}
1362 \DeclareUnicodeComposite{\k}{i}{012F}
1363 \DeclareUnicodeComposite{\.}{I}{0130}
1364 \DeclareUnicodeComposite{\^}{J}{0134}
1365 \DeclareUnicodeComposite{\^}{j}{0135}
1366 \DeclareUnicodeComposite{\^}{j}{0135}
1367 \DeclareUnicodeComposite{\c}{K}{0136}
1368 \DeclareUnicodeComposite{\c}{k}{0137}
1369 \DeclareUnicodeComposite{\'}{L}{0139}
1370 \DeclareUnicodeComposite{\'}{l}{013A}
1371 \DeclareUnicodeComposite{\c}{L}{013B}
1372 \DeclareUnicodeComposite{\c}{l}{013C}
1373 \DeclareUnicodeComposite{\v}{L}{013D}
1374 \DeclareUnicodeComposite{\v}{l}{013E}
1375 \DeclareUnicodeComposite{\'}{N}{0143}
1376 \DeclareUnicodeComposite{\'}{n}{0144}
1377 \DeclareUnicodeComposite{\c}{N}{0145}
1378 \DeclareUnicodeComposite{\c}{n}{0146}
1379 \DeclareUnicodeComposite{\v}{N}{0147}
1380 \DeclareUnicodeComposite{\v}{n}{0148}
1381 \DeclareUnicodeComposite{\=}{O}{014C}
1382 \DeclareUnicodeComposite{\=}{o}{014D}
1383 \DeclareUnicodeComposite{\u}{O}{014E}
1384 \DeclareUnicodeComposite{\u}{o}{014F}
1385 \DeclareUnicodeComposite{\H}{O}{0150}

```

```

1386 \DeclareUnicodeComposite{\H} {o}{0151}
1387 \DeclareUnicodeComposite{\'} {R}{0154}
1388 \DeclareUnicodeComposite{\'} {r}{0155}
1389 \DeclareUnicodeComposite{\c} {R}{0156}
1390 \DeclareUnicodeComposite{\c} {r}{0157}
1391 \DeclareUnicodeComposite{\v} {R}{0158}
1392 \DeclareUnicodeComposite{\v} {r}{0159}
1393 \DeclareUnicodeComposite{\'} {S}{015A}
1394 \DeclareUnicodeComposite{\'} {s}{015B}
1395 \DeclareUnicodeComposite{\^} {S}{015C}
1396 \DeclareUnicodeComposite{\^} {s}{015D}
1397 \DeclareUnicodeComposite{\c} {S}{015E}
1398 \DeclareUnicodeComposite{\c} {s}{015F}
1399 \DeclareUnicodeComposite{\v} {S}{0160}
1400 \DeclareUnicodeComposite{\v} {s}{0161}
1401 \DeclareUnicodeComposite{\c} {T}{0162}
1402 \DeclareUnicodeComposite{\c} {t}{0163}
1403 \DeclareUnicodeComposite{\v} {T}{0164}
1404 \DeclareUnicodeComposite{\v} {t}{0165}
1405 \DeclareUnicodeComposite{\~} {U}{0168}
1406 \DeclareUnicodeComposite{\~} {u}{0169}
1407 \DeclareUnicodeComposite{\=} {U}{016A}
1408 \DeclareUnicodeComposite{\=} {u}{016B}
1409 \DeclareUnicodeComposite{\u} {U}{016C}
1410 \DeclareUnicodeComposite{\u} {u}{016D}
1411 \DeclareUnicodeComposite{\r} {U}{016E}
1412 \DeclareUnicodeComposite{\r} {u}{016F}
1413 \DeclareUnicodeComposite{\H} {U}{0170}
1414 \DeclareUnicodeComposite{\H} {u}{0171}
1415 \DeclareUnicodeComposite{\k} {U}{0172}
1416 \DeclareUnicodeComposite{\k} {u}{0173}
1417 \DeclareUnicodeComposite{\^} {W}{0174}
1418 \DeclareUnicodeComposite{\^} {w}{0175}
1419 \DeclareUnicodeComposite{\^} {Y}{0176}
1420 \DeclareUnicodeComposite{\^} {y}{0177}
1421 \DeclareUnicodeComposite{\"} {Y}{0178}
1422 \DeclareUnicodeComposite{\'} {Z}{0179}
1423 \DeclareUnicodeComposite{\'} {z}{017A}
1424 \DeclareUnicodeComposite{\.} {Z}{017B}
1425 \DeclareUnicodeComposite{\.} {z}{017C}
1426 \DeclareUnicodeComposite{\v} {Z}{017D}
1427 \DeclareUnicodeComposite{\v} {z}{017E}
1428 \DeclareUnicodeComposite{\v} {A}{01CD}
1429 \DeclareUnicodeComposite{\v} {a}{01CE}
1430 \DeclareUnicodeComposite{\v} {I}{01CF}
1431 \DeclareUnicodeComposite{\v} {i}{01D0}
1432 \DeclareUnicodeComposite{\v} {i}{01D0}
1433 \DeclareUnicodeComposite{\v} {O}{01D1}
1434 \DeclareUnicodeComposite{\v} {o}{01D2}
1435 \DeclareUnicodeComposite{\v} {U}{01D3}
1436 \DeclareUnicodeComposite{\v} {u}{01D4}

1437 \DeclareUnicodeComposite{\'} {\AE}{01FC}
1438 \DeclareUnicodeComposite{\'} {\xE}{01FC}

```

```

1439 \DeclareUnicodeComposite{\'} \ae{"01FD}
1440 \DeclareUnicodeComposite{\'} {\æ}{"01FD}
1441 \DeclareUnicodeComposite{\=} \AE{"01E2}
1442 \DeclareUnicodeComposite{\=} {\Ѐ}{"01E2}
1443 \DeclareUnicodeComposite{\=} \ae{"01E3}
1444 \DeclareUnicodeComposite{\=} {\ѧ}{"01E3}
1445 \DeclareUnicodeComposite{\v} \G{"01E6}
1446 \DeclareUnicodeComposite{\v} {\g}{"01E7}
1447 \DeclareUnicodeComposite{\v} {\K}{"01E8}
1448 \DeclareUnicodeComposite{\v} {\k}{"01E9}
1449 \DeclareUnicodeComposite{\k} {\O}{"01EA}
1450 \DeclareUnicodeComposite{\k} {\o}{"01EB}
1451 \DeclareUnicodeComposite{\v} {\j} {"01FO}
1452 \DeclareUnicodeComposite{\v} {\j} {"01FO}
1453 \DeclareUnicodeComposite{\'} \G{"01F4}
1454 \DeclareUnicodeComposite{\'} {\g} {"01F5}
1455 \DeclareUnicodeComposite{\textcommabelow}{S} {"0218}
1456 \DeclareUnicodeComposite{\textcommabelow}{s} {"0219}
1457 \DeclareUnicodeComposite{\textcommabelow}{T} {"021A}
1458 \DeclareUnicodeComposite{\textcommabelow}{t} {"021B}
1459 \DeclareUnicodeComposite{\=} {\Y} {"0232}
1460 \DeclareUnicodeComposite{\=} {\y} {"0233}
1461 \DeclareUnicodeComposite{\.} {\B} {"1E02}
1462 \DeclareUnicodeComposite{\.} {\b} {"1E03}
1463 \DeclareUnicodeComposite{\d} {\B} {"1E04}
1464 \DeclareUnicodeComposite{\d} {\b} {"1E05}
1465 \DeclareUnicodeComposite{\d} {\D} {"1E0C}
1466 \DeclareUnicodeComposite{\d} {\d} {"1E0D}
1467 \DeclareUnicodeComposite{\=} {\G} {"1E20}
1468 \DeclareUnicodeComposite{\=} {\g} {"1E21}
1469 \DeclareUnicodeComposite{\d} {\H} {"1E24}
1470 \DeclareUnicodeComposite{\d} {\h} {"1E25}
1471 \DeclareUnicodeComposite{\d} {\K} {"1E32}
1472 \DeclareUnicodeComposite{\d} {\k} {"1E33}
1473 \DeclareUnicodeComposite{\d} {\L} {"1E36}
1474 \DeclareUnicodeComposite{\d} {\l} {"1E37}
1475 \DeclareUnicodeComposite{\d} {\M} {"1E42}
1476 \DeclareUnicodeComposite{\d} {\m} {"1E43}
1477 \DeclareUnicodeComposite{\d} {\N} {"1E46}
1478 \DeclareUnicodeComposite{\d} {\n} {"1E47}
1479 \DeclareUnicodeComposite{\d} {\R} {"1E5A}
1480 \DeclareUnicodeComposite{\d} {\r} {"1E5B}
1481 \DeclareUnicodeComposite{\d} {\S} {"1E62}
1482 \DeclareUnicodeComposite{\d} {\s} {"1E63}
1483 \DeclareUnicodeComposite{\d} {\T} {"1E6C}
1484 \DeclareUnicodeComposite{\d} {\t} {"1E6D}
1485 \DeclareUnicodeComposite{\d} {\V} {"1E7E}
1486 \DeclareUnicodeComposite{\d} {\v} {"1E7F}
1487 \DeclareUnicodeComposite{\d} {\W} {"1E88}
1488 \DeclareUnicodeComposite{\d} {\w} {"1E89}
1489 \DeclareUnicodeComposite{\d} {\Z} {"1E92}
1490 \DeclareUnicodeComposite{\d} {\z} {"1E93}
1491 \DeclareUnicodeComposite{\d} {\A} {"1EA0}
1492 \DeclareUnicodeComposite{\d} {\a} {"1EA1}

```

```

1493 \DeclareUnicodeComposite{\d} {E}{1EB8}
1494 \DeclareUnicodeComposite{\d} {e}{1EB9}
1495 \DeclareUnicodeComposite{\d} {I}{1ECA}
1496 \DeclareUnicodeComposite{\d} {i}{1ECB}
1497 \DeclareUnicodeComposite{\d} {O}{1ECC}
1498 \DeclareUnicodeComposite{\d} {o}{1ECD}
1499 \DeclareUnicodeComposite{\d} {U}{1EE4}
1500 \DeclareUnicodeComposite{\d} {u}{1EE5}
1501 \DeclareUnicodeComposite{\d} {Y}{1EF4}
1502 \DeclareUnicodeComposite{\d} {y}{1EF5}

1503
```

## 2 Package files

This file now also contains some packages that provide access to the more specialised encodings.

### 2.1 The fontenc package

This package allows authors to specify which encodings they will use. For each encoding `FOO`, the package looks to see if the encoding `FOO` has already been declared. If it has not, the file `fooenc.def` is loaded. The default encoding is set to be `FOO`.

In addition the package at the moment contains extra code to extend the `\@uclclist` (list of upper/lower case pairs) for encodings that involve cyrillic characters. THIS IS A TEMPORARY SOLUTION and will not stay this way forever (or so we hope) but right now we are missing a proper interface for this and didn't wanted to rush it.

```
1504 <*package>
```

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

```

1505 \def\update@uclc@with@cyrilllic{%
1506 \expandafter\def\expandafter@\uclclist\expandafter
1507 {\@uclclist
1508 \cyla\CYRA\cyrabhch\CYRABHCH\cyrabhchdsc\CYRABHCHDSC\cyrabhdze
1509 \CYRABHDZE\cyrabhha\CYRABHHA\cyrae\CYRAE\cyrb\CYRB\cyrbyus
1510 \CYRBYUS\cycr\CYRC\cyrch\CYRCH\cyrchldsc\CYRCHLDSC\cyrchrdsc
1511 \CYRCHRDS\cyrchvcrs\CYRCHVCRS\cyrd\CYRD\cyrdelta\CYRDELTA
1512 \cyrdje\CYRDJE\cyrdze\CYRDZE\cyrdzhe\CYRDZHE\cyre\CYRE\cyreps
1513 \CYREPS\cyrerev\CYREREV\cyrery\CYRERY\cyrf\CYRF\cyrfita
1514 \CYRFITA\cyrg\CYRG\cyrgdsc\CYRGDSC\cyrgdschcrs\CYRGDSCHCRS
1515 \cyrghcrs\CYRGHCRS\cyrghk\CYRGHK\cyrgup\CYRGUP\cyrh\CYRH
1516 \cyrhdsc\CYRHDSC\cyrhhcrs\CYRHHCRS\cyrhhk\CYRHHK\cyrhdsn
1517 \CYRHRDSN\cyri\CYRI\cyrie\CYRIE\cyrii\CYRII\cyrishrt\CYRISHRT
1518 \cyrishrtsc\CYRISHRTDSC\cyrizh\CYRIZH\cyrje\CYRJE\cyrk\CYRK
1519 \cyrkbeak\CYRKBEAK\cyrkdsc\CYRKDSC\cyrkhcrs\CYRKHCRS\cyrkhk
1520 \CYRKHK\cyrkvcrs\CYRKVCRS\cyl\CYRL\cyrlldsc\CYRLDSC\cylrhk
1521 \CYRLHK\cylrhk\CYRLLJE\cylrm\CYRM\cyrmndsc\CYRMNDSC\cyrmhk\CYRMHK
1522 \cyrn\CYRN\cyrndsc\CYRNDSC\cyrng\CYRNG\cyrnhk\CYRNHK\cyrnj
1523 \CYRNJE\cyrnlhk\CYRNLHK\cyro\CYRO\cyrotld\CYROTLD\cyrp\CYRP
1524 \cyrphk\CYRPHK\cyrq\CYRQ\cyrr\CYRR\cyrrdsc\CYRRDSC\cyrphk
1525 \CYRRHK\cyrrtick\CYRRTICK\crys\CYRS\crysacrs\CYRSACRS
1526 \cryscha\CYRSCHWA\crysdesc\CYRSDESC\crysdesc\cyrsemisftsn\CYRSEMISFTSN

```

```

1527 \cyrstfn\CYRSFTSN\cyrsh\CYRSH\cyrshch\CYRSHCH\cyrshha\CYRSHHA
1528 \cyrt\CYRT\cyrtdesc\CYRTDSC\cyrtetse\CYRTETSE\cyrtshe\CYRTSHE
1529 \cyr\CYRU\cyrushrt\CYRUSHRT\cyrv\CYRV\cyrw\CYRW\cqry\CYRY
1530 \carya\CYRYA\caryat\CYRYAT\caryhcrs\CYRYHCRS\caryi\CYRYI\caryo
1531 \CYRYO\caryu\CYRYU\caryz\CYRZ\carydsc\CYRZDSC\caryzh\CYRZH
1532 \caryhdsc\CYRZHDSC}%
1533 \let\update@uclc@with@cyrillic\relax
1534 }

```

Here we process each option:

```

1535 \DeclareOption*{%
1536 \let\encodingdefault\CurrentOption

```

From 2020/02/02 release onward we only load the encoding files if they haven't be loaded already. To check this we look if `\T@encoding` is already defined. If not we load (indicated by setting the switch `@tempswa` to true and we always load if we run in an older format (or rather in a rollback situation).

```

1537 \tempswafalse
1538 \cifl@t@r\fmtversion{2020/02/02}%
1539 {\expandafter\ifx\csname T@\CurrentOption\endcsname\relax
1540 \tempswatrue\fi}%
1541 {\tempswatrue}%

```

Load if necessary:

```

1542 \if@tempswa
1543 \edef\reserved@f{%
1544 \lowercase{\def\noexpand\reserved@f{\CurrentOption enc.def}}%
1545 \reserved@f
1546 \InputIfFileExists\reserved@f
1547 {}{\PackageError{fontenc}%
1548 {Encoding file '\reserved@f' not found.%}
1549 \MessageBreak
1550 You might have misspelt the name of the encoding}%
1551 {Necessary code for this encoding was not
1552 loaded.\MessageBreak
1553 Thus calling the encoding later on will
1554 produce further error messages.}%
1555 \let\reserved@f\relax

```

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

```

1556 \expandafter\in@\expandafter{\CurrentOption}%
1557 {T2A,T2B,T2C,X2,LCY,OT2}%
1558 \ifin@

```

But only if it hasn't already been extended. This might happen if there are several calls to fontenc loading one of the above encodings. If we don't do this check the `\@uclclist` gets unnecessarily big, slowing down the processing at runtime.

```

1559 \expandafter\in@\expandafter\cyra\expandafter
1560 {\@uclclist}%
1561 \ifin@
1562 \else
1563 \update@uclc@with@cyrillic
1564 \fi
1565 \fi

```

```

1566 \fi
1567 }
1568 \ProcessOptions*

```

We select the new font encoding default (i.e., the last encoding specified in the option list. But this encoding may not work with the current `\f@shape`, e.g., LY1 is not defined for `cmr` and therefore packages switching to LY1 usually also change `\rmdefault`. But that only applies at `\begin{document}` so we get a spurious warning if we use what L<sup>A</sup>T<sub>E</sub>X previously used:

```

1569 \%fontencoding\encodingdefault\selectfont

```

So instead we do this here:

```

1570 \usefont\encodingdefault\familydefault\seriesdefault\shapedefault

```

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

```

1571 \let\update@uclc@with@cyrillic\relax

```

Finally we pretend that the `fontenc` package wasn't read in. This allows for using it several times, e.g., in a class file and in the preamble (at the cost of not getting any version info). That kind of hackery shows that using a general purpose package just for loading an encoding is not the right kind of interface for setting up encodings — it will get replaced at some point in the future.

```

1572 \let\@elt\relax
1573 \xdef\@fontenc@load@list{\@fontenc@load@list
1574 \@elt{\csname opt@fontenc.sty\endcsname}}

```

```

1575 \global\expandafter\let\csname ver@@fontenc.sty\expandafter\endcsname
1576 \csname ver@fontenc.sty\endcsname
1577 \global\expandafter\let\csname ver@fontenc.sty\endcsname\relax
1578 \global\expandafter\let\csname opt@fontenc.sty\endcsname\relax
1579 \global\let\@ifl@ter@@\@ifl@ter
1580 \def\@ifl@ter#1#2#3#4#5{\global\let\@ifl@ter\@ifl@ter@@}
1581
```

# File t

## ltcounts.dtx

### 1 Counters and Lengths

Commands for defining and using counters. This file defines:

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

#### 1.1 Environment Counter Macros

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

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

**NOTE:**

`\thefoo` and `\p@foo` must be defined in such a way that `\edef\bar{\thefoo}` or `\edef\bar{\p@foo}` defines `\bar` so that it will evaluate to the counter value at the time of the `\edef`, even after `\foocounter` and any other counters have been changed. This will happen if you use the standard commands `\@arabic`, `\@Roman`, etc.

The following commands are used to define and modify counters.

`\refstepcounter{<foo>}`

Same as `\stepcounter`, but it also defines `\@currentreference` so that a subsequent `\label{<bar>}` command causes `\ref{<bar>}` to generate the current value of counter `<foo>`.

`\@definecounter{<foo>}`

Initializes counter `{<foo>}` (with empty reset list), defines `\p@foo` and `\thefoo` to be null. Also adds `<foo>` to `\cl@ckpt` – the reset list of a dummy counter `@ckpt` used for taking checkpoints for the `\include` system.

`\@addtoreset{<foo>}{<bar>}` : Adds counter `<foo>` to the list of counters `\cl@bar` to be reset when counter `<bar>` is stepped.

`\@removefromreset{<foo>}{<bar>}` : Removes counter `<foo>` to the list of counters `\cl@bar` to be reset when counter `<bar>` is stepped.

`\setcounter{<foo>}{<val>}` : Globally sets `\foocounter` equal to `<val>`.

```
2 \def\setcounter#1#2{%
3 \@ifundefined{c@#1}%
4 {\@nocounterr{#1}}%
5 {\global\csname c@#1\endcsname#2\relax}}
```

(End of 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 of 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 of 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 of definition for `\value`.)

`\@newctr`

```
15 \def\@newctr#1[#2]{%
16 \@ifundefined{c@#2}{\@nocounterr{#2}}{\@addtoreset{#1}{#2}}}
```

(End of 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 of 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 of definition for \@stpelt.)

\cl@@ckpt

```
35 \def\cl@@ckpt{\@elt{page}}
```

(End of 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 of definition for \@definecounter.)

\@addtoreset

```
44 \def\@addtoreset#1#2{\expandafter\@cons\csname cl@#2\endcsname {{#1}}}
```

(End of 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 of 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 of 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 \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 <@ifbothcounters{#1}{#2}{%
120 <@addtoreset{#1}{#2}%
121 \expandafter
122 \gdef\csname the#1\expandafter\endcsname\expandafter
123 \expandafter
124 \csname the#2\expandafter\endcsname\expandafter
125 .\expandafter
126 \arabic\csname c@#1\endcsname}}%
127 <@ifbothcounters{#1}{#2}{%
128 <@ifbothcounters{#1}{#2}{%
129 \let \counterwithout \undefined
130 \let \counterwithout@s \undefined
131 \let \counterwithout@x \undefined
132 \let \counterwithin \undefined
133 \let \counterwithin@s \undefined
134 \let \counterwithin@x \undefined
135 }%
136 }%
137 }
```

(End of 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 of definition for \arabic.)

\roman Representation of *⟨counter⟩* as lower-case Roman numerals.

```
137 \def\roman#1{\expandafter\@roman\csname c@#1\endcsname}
```

(End of definition for \roman.)

\Roman Representation of *⟨counter⟩* as upper-case Roman numerals.

```
138 \def\Roman#1{\expandafter\@Roman\csname c@#1\endcsname}
```

(End of 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 of 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 of 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 of definition for \fnsymbol.)

\@arabic \@arabic\FOOcounter Representation of \FOOcounter as arabic numerals.

```
142 \def\@arabic#1{\number #1} %% changed 29 Apr 86
```

(End of definition for \@arabic.)

\@roman \@roman\FOOcounter Representation of \FOOcounter as lower-case Roman numerals.

```
143 \def\@roman#1{\romannumeral #1}
```

(End of definition for \@roman.)

\@Roman \@Roman\FOOcounter Representation of \FOOcounter as upper-case Roman numerals.

```
144 \def\@Roman#1{\expandafter\@slowromancap\romannumeral #1@}
```

(End of 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
149 \expandafter\@slowromancap
150 \fi
151 }
```

(End of 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 of 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 of definition for \@Alph.)

\@fnsymbol Typesetting old fashioned footnote symbols. This can be done both in text or math mode now.

This macro is another example of an ever recurring problem in TeX: Determining if something is text-mode or math-mode. It is imperative for the decision between text and math to be delayed until the actual typesetting is done as the code in question may go through an \edef or \write where an \ifmmode test would be executed prematurely. Hence in the implementation below, \@fnsymbol is not robust in itself but the parts doing the actual typesetting are.

In the case of \@fnsymbol we make use of the robust command \TextOrMath which takes two arguments and typesets the first if in text-mode and the second if in math-mode. Note that in order for this command to make the correct decision, it must insert a \relax token if run under regular TeX, which ruins any kerning between the preceding characters and whatever awaits typesetting. If you use eTeX as engine for L<sup>A</sup>T<sub>E</sub>X (as recommended) this unfortunate side effect is not present.

```
160 </2ekernel>
161 <|latexrelease|\IncludeInRelease{2015/01/01}{\@fnsymbol}{Use \TextOrMath}%
162 <*2ekernel | latexrelease>
163 \def\@fnsymbol#1{%
164 \ifcase#1\or \TextOrMath{textasteriskcentered}*\or
165 \TextOrMath{textdagger} \dagger\or
166 \TextOrMath{textdaggerdbl} \ddagger\or
167 \TextOrMath{textsection} \mathsection\or
168 \TextOrMath{textparagraph} \mathparagraph\or
169 \TextOrMath{textbardbl} \|\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 of definition for \fnsymbol.)

- \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 of definition for \TextOrMath.)

```
207 〈/2ekernel〉
```

# File u

## ltlength.dtx

### 1 Lengths

```
\newlength Declare #1 to be a new length command.
 \setlength Set the length command, #1, to the value #2.
 \addtolength Increase the value of the length command, #1, by the value #2.
 \settowidth Set the length, #1 to the width of a box containing #2.
 \settoheight Set the length, #1 to the height of a box containing #2.
 \settodepth Set the length, #1 to the depth of a box containing #2.
 1 <*2ekernel>
 2 \message{lengths,}

\newlength
 3 \def\newlength#1{\@ifdefinable#1{\newskip#1}}

(End of 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 of definition for \setlength.)

\addtolength \relax added 24 Mar 86
16 \def\addtolength#1#2{\advance#1 #2\relax}

(End of 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 of definition for \settoheight and others.)
```

`\@settopoint` This macro takes the contents of the skip register that is supplied as its argument and removes the fractional part to make it a whole number of points. This can be used in class files to avoid values like 345.466666pt when calculating a dimension.

```
22 \def\@settopoint#1{\divide#1\p@\multiply#1\p@}
23 \end{macro}
```

(*End of definition for `\@settopoint`.*)

# File v

## ltfssbas.dtx

This file contains the main implementation of the ‘low level’ font selection commands. See other parts of the L<sup>A</sup>T<sub>E</sub>X distribution, or *The L<sup>A</sup>T<sub>E</sub>X Companion* for higher level documentation of the L<sup>A</sup>T<sub>E</sub>X ‘New’ Font Selection Scheme.

**Warning:** The macro documentation is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

### 1 Preliminary macros

We define a number of macros that will be used later.

`\@nomath` `\@nomath` is used by most macros that will have no effect in math mode. It issues a warning message.

```
1 {*2ekernel}
2 \def\@nomath#1{\relax\ifmmode
3 \@font@warning{Command \noexpand#1invalid in math mode}\fi}
```

(End of definition for `\@nomath`.)

`\no@alphabet@error` The macro `\no@alphabet@error` is called whenever the user requests a math *alphabet* that is not available in the current *version*. In math mode an error message is produced otherwise the command keeps silent. The argument is the name of the control sequence that identifies the math *alphabet*. The `\relax` at the beginning is necessary to prevent T<sub>E</sub>X from scanning too far in certain situations.

```
4 \gdef\no@alphabet@error#1{\relax\ifmmode
5 \@latex@error{Math\space alphabet\space identifier\space
6 \noexpand#1is\space undefined\space in\space math\space
7 version\space ‘\math@version’}%
8 {Your\space requested\space math\space alphabet\space
9 is\space undefined\space in\space the\space current\space
10 math\space version.^^JCheck\space the\space spelling\space
11 or\space use\space the\space \noexpand\SetMathAlphabet\space
12 command.}%
13 \fi}
```

(End of definition for `\no@alphabet@error`.)

`\new@mathgroup` `\mathgroup` We also give a new name to `\newfam` and `\fam` to avoid verbal confusion (see the introduction).<sup>27</sup>

```
14 \%def\new@mathgroup{\alloc@8\mathgroup\chardef\sixt@n}
15 \let\mathgroup\fam
16 \%let\newfam\new@mathgroup
17 \onlypreamble\new@mathgroup
```

(End of definition for `\new@mathgroup` and `\mathgroup`.)

<sup>27</sup>For the same reason it seems advisable to `\let\fam` and `\newfam` equal to `\relax`, but this is commented out to retain compatibility to existing style files.

## 2 Macros for setting up the tables

```
\DeclareFontShape{The macro \DeclareFontShape takes 6 arguments:
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 of 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 of 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 of 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 of 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 of definition for `\DeclareFontFamily`.)

`\cdp@list` We initialize the code page list to be empty.

```
114 \let\cdp@list\@empty
115 \onlypreamble\cdp@list
```

(End of definition for `\cdp@list`.)

```
\cdp@elt
```

```
116 \let\cdp@elt\relax
117 \onlypreamble\cdp@elt
```

(End of 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 of definition for `\DeclareFontEncoding`.)

`\LastDeclaredEncoding` The last encoding being declared by `\DeclareFontEncoding`.

```
140 \def\LastDeclaredEncoding{}%
```

(End of 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 of 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 of definition for \DeclareFontEncodingDefaults.)

#### \default@T

#### \default@M

```
183 \let\default@T\empty
184 \let\default@M\empty
```

(End of definition for \default@T and \default@M.)

#### \DeclarePreloadSizes

```
185 \def\DeclarePreloadSizes#1#2#3#4#5{%
186 \@ifundefined{T@#1}{%
187 {\@latex@error{Encoding scheme '#1' unknown}\@eha}%
188 }%
```

Don't know at the moment what this group here does!

```
189 \begingroup
```

We define a macro `\reserved@f`<sup>28</sup> that grabs the next *size* and loads the corresponding font. This is done by delimiting `\reserved@f`'s only argument by the token `,` (comma).

```
190 \def\reserved@f##1,{%
```

The end of the list will be detected when there are no more elements, i.e. when `\reserved@f`'s argument is empty. The trick used here is explained in Appendix D of the TeXbook: if the argument is empty the `\if` will select the first clause and `\let` `\reserved@f` equal to `\relax`. (We use the `>` character here since it cannot appear in font file names.)

```
191 \if>##1>%
192 \let\reserved@f\relax
193 \else
```

Otherwise, we define `\font@name` appropriately and call `\pickup@font` to do the work. Note that the requested `\curr@fontshape` combination must have been defined, or you will get an error. The definition of `\font@name` is carried out globally to be consistent with the rest of the code in this file.

```
194 \xdef\font@name{\csname#1/#2/#3/#4/#1\endcsname}%
195 \pickup@font
```

Now we forget the name of the font just loaded. More precisely, we set the corresponding control sequence to `\relax`. This means that later on, when the font is first used, the macro `\define@newfont` is called again to execute the 'extra' macro for this font.

```
196 \global\expandafter\let\font@name\relax
197 \fi
```

Finally we call `\reserved@f` again to process the next *size*. If `\reserved@f` was `\let` equal to `\relax` this will end the macro.

```
198 \reserved@f}%
```

We finish with reinserting the list of sizes after the `\reserved@f` macro and appending an empty element so that the end of the list is recognized properly.

```
199 \reserved@f#5,,%
200 \endgroup
201 }%
202 }
203 \onlypreamble\DeclarePreloadSizes
```

(End of 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 of definition for `\ifmath@fonts`.)

---

<sup>28</sup>We cannot use `\@tempa` since it is needed in `\pickup@font`.

\DeclareMathSizes \DeclareMathSizes takes the text size, math text size, math script size, and math scriptscript size as arguments and defines the right \S@... macro.

```

205 \def\DeclareMathSizes{%
206 \@ifstar{\@DeclareMathSizes\math@fontsfalse}{%
207 {\@DeclareMathSizes{}}}
208 \onlypreamble\DeclareMathSizes

```

(End of 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 of 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 of 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 of definition for \@@enc@update.)*

`\enc@update` The default action in `\selectfont` is to do nothing.

```
278 \let\enc@update\relax
```

*(End of 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 of 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 of 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 of 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 of 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 of definition for \f@linespread.*)

**\cf@encoding**

```

326 \let\f@encoding\empty
327 \let\cf@encoding\empty

```

(*End of 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 of definition for \@defaultunits.)
- \strip@pt This macro strips the characters pt produced by using \the on a dimen register.
- ```
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 of 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
```
- If there has been a frozen math version reset locally. See GH 1028.
- ```
340   \ifcsname mv@\math@version @frozen\endcsname
341     \expandafter\let
342     \csname mv@\math@version @frozen\expandafter\endcsname
343     \csname mv@\math@version\endcsname
344   \fi
345   \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.
- ```
346 \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.
- ```
347   \aftergroup\glb@settings
348   \fi}
```

(End of 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 TeX function must be hidden.

Anyway, since we don't have this and won't have it in L^AT_EX 2_C 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.

```
\frozen@everymath  New internal names for \everymath and \everydisplay.  
\frozen@everydisplay  
349  \let\frozen@everymath\everymath  
350  \let\frozen@everydisplay\everydisplay
```

(End of definition for `\frozen@everymath` and `\frozen@everydisplay`.)

```
\everymath      Now we provide now user hooks that will be called in the frozen internals.  
\everydisplay  
351  \newtoks\everymath  
352  \newtoks\everydisplay
```

(End of definition for `\everymath` and `\everydisplay`.)

```
\frozen@everydisplay Now we define the behaviour of the frozen hooks: first check the math setup then call the user hook.
```

The check code may push tokens after the math formula with `\aftergroup` and they would prevent a \$\$ from dropping following spaces. We therefore use a switch to be set as the first thing after the group so that following code can determine if there was a display or some inline math (in the latter case we better not drop spaces). After setting the switch we also have to place `\ignorespaces` because setting the switch may be the only thing that happens after the display. The issue with handling of spaces was found in 2022, but it is really a bug fix for the code added in 2021/11.

```
353  </2ekernel>  
354  <latexrelease>\IncludeInRelease{2021/11/15}  
355  <latexrelease>  {\frozen@everydisplay}{Handle spaces after math} %  
356  <*2ekernel | latexrelease>  
357  \frozen@everydisplay = {  
358    \aftergroup\@ignoretrue \aftergroup\ignorespaces  
359    \check@mathfonts  
360    \the\everydisplay}
```

(End of definition for `\frozen@everydisplay`.)

```
\frozen@everymath The frozen code for inline math is similar, except that here we do not want to drop following spaces.
```

```
361  \frozen@everymath = {  
362    \aftergroup\@ignorefalse  
363    \check@mathfonts  
364    \the\everymath}
```

```

(End of definition for \frozen@everymath.)
```

```

365  </2ekernel | latexrelease>
366  <latexrelease>\EndIncludeInRelease
367  <latexrelease>\IncludeInRelease{2020/10/01}
368  <latexrelease>  {\frozen@everydisplay}{Handle spaces after math}%
369  <latexrelease>
370  <latexrelease>\frozen@everydisplay = {\check@mathfonts
371  <latexrelease>                                \the\everydisplay}
372  <latexrelease>\frozen@everymath = {\check@mathfonts
373  <latexrelease>                                \the\everymath}
374  <latexrelease>
375  <latexrelease>\EndIncludeInRelease
376  <*2ekernel>
```

\curr@math@size This holds locally the current math size.

```
377 \let\curr@math@size\empty
```

```
(End of definition for \curr@math@size.)
```

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.

```

378 \def\pickup@font{%
379   \expandafter \ifx \font@name \relax
380     \define@newfont
381   \fi}
```

```
(End of 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

```
382 {\catcode`/=\catcode`/}
```

and define \split@name so that it will define our private \f@encoding, \f@family, \f@series, \f@shape, and \f@size macros.

```

383 \gdef\split@name#1/#2/#3/#4/#5\@nil{\def\f@encoding{#1}%
384   \def\f@family{#2}%
385   \def\f@series{#3}%
386   \def\f@shape{#4}%
387   \def\f@size{#5}}}
```

```
(End of definition for \split@name.)
```

\curr@fontshape Abbreviation which may get removed again for speed.

```
388 \def\curr@fontshape{\f@encoding/\f@family/\f@series/\f@shape}
```

```
(End of definition for \curr@fontshape.)
```

```
\define@newfont Now we can tackle the problem of defining a new font.
```

```
389 \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.

```
390 \begingroup
391   \let\typeout\@font@info
392   \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`.

```
393   \expandafter\expandafter\expandafter
394     \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.

```
395 %   \expandafter\ifx
396 %     \csname\curr@fontshape\endcsname \relax
397   \try@load@fontshape % try always
398 %   \fi
399   \expandafter\ifx
400     \csname\curr@fontshape\endcsname \relax
401   \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).

```
402 %   \csname\curr@fontshape\endcsname
403   \extract@font\fi
```

We are nearly finished and must only restore the `\escapechar` by closing the group.

```
404 \endgroup}
405 \def\try@load@fontshape{%
406   \expandafter
407   \ifx\csname \f@encoding+\f@family\endcsname\relax
408     \@font@info{Trying to load font information for
409       \f@encoding+\f@family}%

```

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.

```
410 \global\expandafter\let
411   \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).

```
412   \nfss@catcodes
413   \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.

```

414     \edef\reserved@a{%
415         \lowercase{%
416             \noexpand\InputIfFileExists{\f@encoding\f@family.fd}}{}}%
417         \reserved@a\relax
418         {\@input{\f@encoding\f@family.fd}}{}}%
419     \fi}

```

(End of 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.

```
420 \def\nfss@catcodes{%
```

We start by making @ a letter and ignoring all blanks and newlines.

```

421     \makeatletter
422     \catcode`\@=11
423     \catcode`^I=12
424     \catcode`^M=13

```

Then we set up \, {, }, # and % in case an `.fd` file is loaded during a verbatim environment.

```

425     \catcode`\\=0
426     \catcode`{\=1
427     \catcode`}=2
428     \catcode`#=6
429     \catcode`^=7
430     \catcode`^@=14

```

The we make sure that the important syntax parts have the right `\catcode`.

```

431     \@makeother<=0
432     \@makeother>=1
433     \@makeother*=2
434     \@makeother.=3
435     \@makeother-=4
436     \@makeother/=5
437     \@makeother[=6
438     \@makeother]=7
439     \@makeother`=8
440     \@makeother'=9
441     \@makeother"=10
442 }

```

(End of definition for `\nfss@catcodes`.)

`\LoadFontDefinitionFile` Load and .fd files for some encoding and family (if it exists).

```
443 </2ekernel>
444 <*2ekernel | latexrelease>
445 <latexrelease>\IncludeInRelease{2020/02/02}%
446 <latexrelease>          {\LoadFontDefinitionFile}{Loading .fd files}%
447 \def\LoadFontDefinitionFile#1#2{%
448   \begingroup
449     \edef\f@encoding{#1}%
450     \edef\f@family{#2}%
451     \try@load@fontshape
452   \endgroup
453 }
454 </2ekernel | latexrelease>
455 <latexrelease>\EndIncludeInRelease
456 <latexrelease>\IncludeInRelease{0000/00/00}%
457 <latexrelease>          {\LoadFontDefinitionFile}{Loading .fd files}%
458 <latexrelease>
459 <latexrelease>\let\LoadFontDefinitionFile@\undefined
460 <latexrelease>\EndIncludeInRelease
461 <*2ekernel>
```

(End of 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 uidot 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).

```
462 </2ekernel>
463 <*2ekernel | latexrelease>
464 <latexrelease>\IncludeInRelease{2020/02/02}%
465 <latexrelease>          {\DeclareFontFamilySubstitution}{Provide family substitution}%
466 \begingroup
467 \nfss@catcodes
468 \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.

```
469   \LoadFontDefinitionFile{#1}{#2}%
470   \LoadFontDefinitionFile{#1}{#3}%
```

```

471 \DeclareFontShape{\#1}{\#2}{m}{it}{<->ssub * #3/m/it}{}%
472 \DeclareFontShape{\#1}{\#2}{m}{n}{<->ssub * #3/m/n}{}%
473 \DeclareFontShape{\#1}{\#2}{m}{sc}{<->ssub * #3/m/sc}{}%
474 \DeclareFontShape{\#1}{\#2}{m}{s1}{<->ssub * #3/m/s1}{}%

```

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.

```

475 \DeclareFontShape{\#1}{\#2}{m}{sw}{<->ssub * #3/m/sw}{}%
476 \DeclareFontShape{\#1}{\#2}{m}{scit}{<->ssub * #3/m/scit}{}%
477 \DeclareFontShape{\#1}{\#2}{m}{scsl}{<->ssub * #3/m/scsl}{}%

```

Same game with b and bx, for other weights you are on your own:

```

478 \DeclareFontShape{\#1}{\#2}{b}{it}{<->ssub * #3/b/it}{}%
479 \DeclareFontShape{\#1}{\#2}{b}{n}{<->ssub * #3/b/n}{}%
480 \DeclareFontShape{\#1}{\#2}{b}{scit}{<->ssub * #3/b/scit}{}%
481 \DeclareFontShape{\#1}{\#2}{b}{scsl}{<->ssub * #3/b/scsl}{}%
482 \DeclareFontShape{\#1}{\#2}{b}{sc}{<->ssub * #3/b/sc}{}%
483 \DeclareFontShape{\#1}{\#2}{b}{s1}{<->ssub * #3/b/s1}{}%
484 \DeclareFontShape{\#1}{\#2}{b}{sw}{<->ssub * #3/b/sw}{}%
485 \DeclareFontShape{\#1}{\#2}{bx}{it}{<->ssub * #3/bx/it}{}%
486 \DeclareFontShape{\#1}{\#2}{bx}{n}{<->ssub * #3/bx/n}{}%
487 \DeclareFontShape{\#1}{\#2}{bx}{scit}{<->ssub * #3/bx/scit}{}%
488 \DeclareFontShape{\#1}{\#2}{bx}{scsl}{<->ssub * #3/bx/scsl}{}%
489 \DeclareFontShape{\#1}{\#2}{bx}{sc}{<->ssub * #3/bx/sc}{}%
490 \DeclareFontShape{\#1}{\#2}{bx}{s1}{<->ssub * #3/bx/s1}{}%
491 \DeclareFontShape{\#1}{\#2}{bx}{sw}{<->ssub * #3/bx/sw}{}%
492 }
493 \endgroup
494 </2ekernel | latexrelease>
495 <latexrelease>\EndIncludeInRelease
496 <latexrelease>\IncludeInRelease{0000/00/00}%
497 <latexrelease> {\DeclareFontFamilySubstitution}{Provide family substitution}%
498 <latexrelease>
499 <latexrelease>\let\DeclareFontFamilySubstitution@\undefined
500 <latexrelease>\EndIncludeInRelease
501 <*2ekernel>

```

(*End of 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!

```

502 </2ekernel>
503 <*2ekernel | latexrelease>
504 <latexrelease>\IncludeInRelease{2019/10/01}%
505 <latexrelease> {\DeclareErrorFont}{No side effects please}%
506 \def\DeclareErrorFont#1#2#3#4#5{%
507   \xdef\error@fontshape{%
508     \noexpand\expandafter\noexpand\split@name\noexpand\string
509     \expandafter\noexpand\csname#1/#2/#3/#4/#5\endcsname
510     \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.

```

511 %      \gdef\f@encoding{#1}%
512      \gdef\default@family{#2}%
513      \gdef\default@series{#3}%
514      \gdef\default@shape{#4}%
515 }
516 </2ekernel | latexrelease>
517 <latexrelease>\EndIncludeInRelease
518 <latexrelease>\IncludeInRelease{0000/00/00}%
519 <latexrelease>          {\DeclareErrorFont}{No side effects please}%
520 <latexrelease>
521 <latexrelease>\def\DeclareErrorFont#1#2#3#4#5{%
522 <latexrelease>      \xdef\error@fontshape{%
523 <latexrelease>          \noexpand\expandafter\noexpand\split@name\noexpand\string
524 <latexrelease>          \expandafter\noexpand\csname#1/#2/#3/#4/#5\endcsname
525 <latexrelease>          \noexpand\@nil}%
526 <latexrelease>      \gdef\default@family{#2}%
527 <latexrelease>      \gdef\default@series{#3}%
528 <latexrelease>      \gdef\default@shape{#4}%
529 <latexrelease>      \global\let\f@family\default@family
530 <latexrelease>      \global\let\f@series\default@series
531 <latexrelease>      \global\let\f@shape\default@shape
532 <latexrelease>      \gdef\f@size{#5}%
533 <latexrelease>      \gdef\f@baselineskip{#5pt}%
534 <latexrelease>}
535 <latexrelease>\EndIncludeInRelease
536 <*2ekernel>
537 \onlypreamble\DeclareErrorFont

```

(*End of 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 T_EX will go into an infinite loop. But if the defaults are set incorrectly one deserves nothing else!

```

538 </2ekernel>
539 <latexrelease>\IncludeInRelease{2015/01/01}{\wrong@fontshape}%
540 <latexrelease>          {Font substitution in preamble}%
541 <*2ekernel | latexrelease>
542 \def\wrong@fontshape{%
543   \csname D@\f@encoding\endcsname % install defaults if in math

```

We remember the wanted `\curr@fontshape` combination which we will need in a moment.

```

544 \edef\reserved@a{\csname\curr@fontshape\endcsname}%
545 \ifx\last@fontshape\reserved@a
546   \errmessage{Corrupted NFSS tables}%
547   \error@fontshape
548 \else

```

Then we warn the user about the mess and set the shape to its default.

```

549 \let\f@shape\default@shape

```

If the combination is not known, try the default *series*.

```
550     \expandafter\ifx\csname\curr@fontshape\endcsname\relax
551         \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!

```
552     \expandafter
553     \ifx\csname\curr@fontshape\endcsname\relax
554         \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}.

```
555     \begingroup
556         \try@load@fontshape
557     \endgroup
558     \fi \fi
559     \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.

```
560     \@font@warning{Font shape `'\expandafter\string\reserved@a'
561                     \expandafter@\gobble\string`\undefined\MessageBreak
562                     using '\curr@fontshape' instead \@wrong@font@char}%
563     \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.

```
564     \gdef\@defaultsubs{%
565         \@font@warning{Some font shapes were not available, defaults
566                     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.

```
567     \global\expandafter\expandafter\expandafter\let
568         \expandafter\reserved@a
569         \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.

```

570      \xdef\font@name{%
571          \csname\curr@fontshape/\f@size\endcsname}%
572          \pickup@font}
573  </2ekernel | latexrelease>
574  <latexrelease>\EndIncludeInRelease
575  <latexrelease>\IncludeInRelease{0000/00/00}{\wrong@fontshape}%
576  <latexrelease>          {Font substitution in preamble}%
577  <latexrelease>\def\wrong@fontshape{%
578  <latexrelease>          \csname D@\f@encoding\endcsname
579  <latexrelease>          \edef\reserved@a{\csname\curr@fontshape\endcsname}%
580  <latexrelease>          \ifx\last@fontshape\reserved@a
581  <latexrelease>          \errmessage{Corrupted NFSS tables}%
582  <latexrelease>          \error@fontshape
583  <latexrelease> \else
584  <latexrelease>          \let\f@shape\default@shape
585  <latexrelease>          \expandafter\ifx\csname\curr@fontshape\endcsname\relax
586  <latexrelease>          \let\f@series\default@series
587  <latexrelease>          \expandafter
588  <latexrelease>          \ifx\csname\curr@fontshape\endcsname\relax
589  <latexrelease>          \let\f@family\default@family
590  <latexrelease>          \fi \fi
591  <latexrelease> \fi
592  <latexrelease> \c@font@warning{Font shape
593  <latexrelease>          ‘\expandafter\string\reserved@a’
594  <latexrelease>          \expandafter\@gobble\string\@undefined
595  <latexrelease>          \MessageBreak
596  <latexrelease>          using ‘\curr@fontshape’ instead\@wrong@font@char}%
597  <latexrelease> \global\let\last@fontshape\reserved@a
598  <latexrelease> \gdef\@defaultsubs{%
599  <latexrelease>          \c@font@warning{Some font shapes were not available,
600  <latexrelease>          defaults substituted.\@gobbletwo}}%
601  <latexrelease> \global\expandafter\expandafter\expandafter\let
602  <latexrelease>          \expandafter\reserved@a
603  <latexrelease>          \csname\curr@fontshape\endcsname
604  <latexrelease>          \xdef\font@name{%
605  <latexrelease>          \csname\curr@fontshape/\f@size\endcsname}%
606  <latexrelease>          \pickup@font}
607  <latexrelease>\EndIncludeInRelease
608  {*2ekernel}

```

(End of 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.

```
609 \let\@wrong@font@char\@empty
```

(End of definition for `\@wrong@font@char`.)

`\@@defaultsubs` See above.

```
610 \let\@defaultsubs\relax
```

(End of 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).

611 `\def\strip@prefix#1>{}`

(End of 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.

612 `\let\install@mathalphabet\gdef`

(End of definition for `\install@mathalphabet`.)

`\math@fonts`

613 `\let\math@fonts\empty`

(End of 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.

614 `%\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.

615 `% \begingroup`

We set the math fonts for the *family* in question by calling `\getanddefine@fonts` in the correct environment.

616 `% \escapechar\m@ne`

617 `% \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%`

We globally select the math fonts...

618 `% \globaldefs\m@ne \math@fonts`

... and close the group to restore `\globaldefs` and `\escapechar`.

619 `% \endgroup`

As long as no *size* or *version* change occurs the *math alphabet identifier* should simply switch to the installed *math group* instead of calling `\select@group` unnecessarily. So we globally redefine the first argument (the new *math alphabet identifier*) 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 *math alphabet identifier* and then calling the new definition to switch to the wanted *math group*. Now we define the *math alphabet identifier* as a call to the `\use@mathgroup` command.

620 `% \xdef#1{\noexpand\use@mathgroup\noexpand#2%`

621 `{\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.

```
622 %     \expandafter\extract@alph@from@version
623 %         \csname mv@\math@version\expandafter\endcsname
624 %         \expandafter{\number\csname c@mv@\math@version\endcsname}%
625 %         #1%
626 %         \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.

```
627 %\expandafter #1\fi}
```

(End of 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.

```
628 \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`.

```
629 \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`.

```
630 \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.

```
631 \def\reserved@c####1{\gdef#1{##1####1##3}}%
```

Then we execute our auxiliary macro.

632 \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.

633 \def\reserved@a\select@group#3##1##2\@nil{%

This macro can now directly rebuild the math version definition by calling \reserved@c:

```
634     \reserved@c{%
635         \getanddefine@fonts{#2}##2%
636         \install@mathalphabet#3{%
637             \relax\ifmmode \else \non@alpherr#3\fi
638             \use@mathgroup##1{#2}}}%
```

In addition it defines the alphabet the way it should be used from now on.

```
639 \gdef#3{\relax\ifmmode \else \non@alpherr#3\fi  
640 \use@mathgroup##1{#2}}%
```

Finally, we only have to call this macro `\reserved@a` on the old definitions recorded in `\reserved@b`:

```
641     \expandafter\reserved@a\reserved@b\@nil  
642 }
```

(End of definition for \extract@alpha@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.

```
643 \let\math@bgroup\bgroup  
644 \def\math@egroup{\#1{#1\egroup}}
```

(End of 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`.

```

645 \gdef\calculate@math@sizes{%
646   \font@info{Calculating\space math\space sizes\space for\space
647   size\space <\f@size>}%
648   \dimen@\f@size \p@
649   \tempdima \defaultscriptratio \dimen@
650   \dimen@ \defaultscriptsingleratio \dimen@
651   \expandafter\xdef\csname S@\f@size\endcsname{%
652     \gdef\noexpand\math@font{%
653       \gdef\noexpand\math@font{%
654         \gdef\noexpand\math@font{%
655           \noexpand\math@fonttrue}}}

```

(End of 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.

```

656 \def\defaultscriptratio{.7}
657 \def\defaultscriptscriptratio{.5}

```

(End of definition for \defaultscriptratio and \defaultscriptscriptratio.)

\noaccents@ If we don't have a definition for \noaccents@ we provide a dummy.

```

658 \ifx\noaccents@\undefined
659   \let\noaccents@\empty
660 \fi

```

(End of 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.

```

661 </2ekernel>
662 <latexrelease>\IncludeInRelease{2017/01/01}{\showhyphens}%
663 <latexrelease>                                {XeTeX support for \showhyphens}%
664 <*2ekernel | latexrelease>
665 \ifx\XeTeXcharclass\undefined

```

Version for engines other than XeT_EX.

```

666 \DeclareRobustCommand\showhyphens[1]{%
667   \setbox0\vbox{%
668     \color@begingroup
669     \everypar{}%
670     \parfillskip\z@skip\hsize\maxdimen
671     \normalfont
672     \pretolerance\m@ne\tolerance\m@ne\hbadness\z@\showboxdepth\z@\
673     \color@endgroup}}

```

```

674 \else

```

XeT_EX version. When using system fonts XeT_EX 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.

```

675 \DeclareRobustCommand\showhyphens[1]{%
676   \setbox0\vbox{%
677     \usefont{TU}{lmr}{m}{n}%
678     \hsize 1sp %
679     \hbadness\OM
680     \hfuzz\maxdimen
681     \tracingonline\z@%
682     \everypar={}%
683     \leftskip\z@skip
684     \rightskip\z@skip
685     \parfillskip\z@skip

```

```

686   \hyphenpenalty=-\@M
687   \pretolerance\m@ne
688   \interlinepenalty\z@
689   \clubpenalty\z@
690   \widowpenalty\z@
691   \brokenpenalty1127 %
692   \setbox\z@\hbox{}%
693   \noindent
694   \hskip\z@skip
695   #1%
696   \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`).

```

697   \loop
698   @tempswafalse
699   \ifnum\lastnodetype=11\unskip\@tempswatrue\fi
700   \ifnum\lastnodetype=12\unkern\@tempswatrue\fi
701   \ifnum\lastnodetype=13 %
702     \count@\lastpenalty
703     \unpenalty\@tempswatrue
704   \fi
705   \ifnum\lastnodetype=\@ne
706     \setbox\tw@\lastbox\@tempswatrue
707     \setbox0\hbox{\unhbox\tw@\unskip\unskip\unpenalty
708       \ifnum\count@=1127 \else\ \fi
709       \unhbox0}%
710     \count@\z@
711   \fi
712   \if@tempswa
713     \repeat
714   \hbadness\z@
715   \hsize\maxdimen
716   \showboxdepth\z@
717   \tolerance\m@ne
718   \hyphenpenalty\z@
719   \noindent\unhbox\z@
720 }
721 \fi
722 </2ekernel | latexrelease>
723 <latexrelease>\EndIncludeInRelease
724 <latexrelease>\IncludeInRelease{0000/00/00}{\showhyphens}%
725 <latexrelease>          {XeTeX support for \showhyphens}%
726 <latexrelease>\gdef\showhyphens#1{%
727 <latexrelease>  \setbox0\vbox{%
728 <latexrelease>    \color@begingroup
729 <latexrelease>    \everypar{}%
730 <latexrelease>    \parfillskip\z@skip\hsize\maxdimen
731 <latexrelease>    \normalfont
732 <latexrelease>    \pretolerance\m@ne\tolerance\m@ne
733 <latexrelease>    \hbadness\z@\showboxdepth\z@\ #1%
734 <latexrelease>    \color@endgroup}%
735 <latexrelease>\EndIncludeInRelease
736 <*2ekernel>

```

(End of definition for \showhyphens.)

\addto@hook We need a macro to add tokens to a hook.

737 \long\def\addto@hook#1#2{\#1\expandafter{\the#1#2}}

(End of definition for \addto@hook.)

\@vpt

738 \def\@vpt{5}

(End of definition for \@vpt.)

\@vipt

739 \def\@vipt{6}

(End of definition for \@vipt.)

\@viiipt

740 \def\@viiipt{7}

(End of definition for \@viiipt.)

\@viiiippt

741 \def\@viiiippt{8}

(End of definition for \@viiiippt.)

\@ixpt

742 \def\@ixpt{9}

(End of definition for \@ixpt.)

\@xipt

743 \def\@xipt{10}

(End of definition for \@xipt.)

\@xiipt

744 \def\@xiipt{10.95}

(End of definition for \@xiipt.)

\@xiiipt

745 \def\@xiiipt{12}

(End of definition for \@xiiipt.)

\@xivpt

746 \def\@xivpt{14.4}

(End of definition for \@xivpt.)

\@xviipt

747 \def\@xviipt{17.28}

(End of definition for \@xviipt.)

```
\@xxpt
748 \def\@xxpt{20.74}
(End of definition for \@xxpt.)  

\@xxvpt
749 \def\@xxvpt{24.88}
(End of definition for \@xxvpt.)
750 ⟨/2ekernel⟩
```

File w

ltfssaxes.dtx

This file contains the implementation for handling extra axes splitting the series and the values into sub-categories. selection commands. See other parts of the L^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 of definition for `\DeclareFontSeriesChangeRule`.)

1.2 Mapping rules for series changes

The rules set up use explicit series values not `\..default` indirections; my current feeling is that this is in fact better.

With 9 weights and 9 width classes this table is getting a bit large in the end (324 entries) but on the other hand it doesn't change and accessing speed and it is fast this way.

We could alternatively split the axis and maintain weight and width separately, but that would take more processing time and would not allow for setting up explicit exceptions nicely (not sure that this would ever get used though).

Design considerations for mapping entries:

- We make `m` to reset both weight and width (as this is how it always worked). To reset just the width `?m` is provided and to reset just the weight `m?`.
- We do support “`mwidth`” and “`weightm`”, e.g., `mec` to mean “go to medium weight and extra-condensed width”. At the end of the process we automatically drop any leftover `m` in the series name (unless it is just a single `m`).
- If there is no table entry then the target series is used unconditionally. This means that any request to set both weight and width (e.g. `bx` or `ulc`) needs no table entries. For that reason there are no entries which have a weight+width as request (i.e., second argument).

In particular this is also true for cases involving `m`, e.g., `bm` (bold medium width) which automatically gets reduced result in `b` or `mc` (medium weight condensed) which becomes `c` as a result.

- Only a few entries have “alternative” values and perhaps most of them should get dropped. Or maybe not ... needs some thought perhaps.

The idea is that you don't want the normal substitution to kick in because that would reset the shape first and it may be better to stay with `b` when a change to `c` is requested and `bc` doesn't exist, than to go to first change the shape to `n` and then find that `bc/n` doesn't exist either and thus ending up with `m/n`.

- Also: while I did set up all nine standard weight values from `ul` to `ub` I only bothered to provide entries for `ec`, `sc`, `c` and `x`, because other levels of compression/expansion are not in any real fonts that I know.

Could and perhaps should be eventually extended to cover the whole set.

```
6 \DeclareFontSeriesChangeRule {bc}{b}{bc}{}  
7 \DeclareFontSeriesChangeRule {bc}{c}{bc}{}  
8 \DeclareFontSeriesChangeRule {bc}{eb}{ebc}{}  
9 \DeclareFontSeriesChangeRule {bc}{ec}{bec} {bc}  
10 \DeclareFontSeriesChangeRule {bc}{el}{elc}{}  
11 \DeclareFontSeriesChangeRule {bc}{l}{lc}{}  
12 \DeclareFontSeriesChangeRule {bc}{sb}{sbc}{}  
13 \DeclareFontSeriesChangeRule {bc}{sc}{bsc} {bc}  
14 \DeclareFontSeriesChangeRule {bc}{s1}{slc}{}  
15 \DeclareFontSeriesChangeRule {bc}{ub}{ubc}{}  
16 \DeclareFontSeriesChangeRule {bc}{ul}{ulc}{}  
17 \DeclareFontSeriesChangeRule {bc}{x}{bx}{}  
18 \end{document}
```

```

18 \DeclareFontSeriesChangeRule {bx}{b}{bx}{}%
19 \DeclareFontSeriesChangeRule {bx}{c} {bc} {bx} %<-----
20 \DeclareFontSeriesChangeRule {bx}{eb}{ebx}{}%
21 \DeclareFontSeriesChangeRule {bx}{ec} {bec} {bx} %<-----
22 \DeclareFontSeriesChangeRule {bx}{el}{elx}{}%
23 \DeclareFontSeriesChangeRule {bx}{l}{lx}{}%
24 \DeclareFontSeriesChangeRule {bx}{sb} {sbx} {}%
25 \DeclareFontSeriesChangeRule {bx}{sc} {bsc} {bx} %<-----
26 \DeclareFontSeriesChangeRule {bx}{sl}{slx} {}%
27 \DeclareFontSeriesChangeRule {bx}{ub}{ubx}{}%
28 \DeclareFontSeriesChangeRule {bx}{ul}{ulx}{}%
29 \DeclareFontSeriesChangeRule {bx}{x}{bx}{}%

30 \DeclareFontSeriesChangeRule {b}{bx} {bx} {b} %<-----
31 \DeclareFontSeriesChangeRule {b}{c} {bc} {b} %<-----
32 \DeclareFontSeriesChangeRule {b}{ec} {bec} {b} %<-----
33 \DeclareFontSeriesChangeRule {b}{sb} {sb} {b} %<-----
34 \DeclareFontSeriesChangeRule {b}{sc} {bsc} {b} %<-----
35 \DeclareFontSeriesChangeRule {b}{x} {bx} {b} %<-----

36 \DeclareFontSeriesChangeRule {c}{bx} {bx} {b} %<-----
37 \DeclareFontSeriesChangeRule {c}{b}{bc}{}%
38 \DeclareFontSeriesChangeRule {c}{eb}{ebc}{}%
39 \DeclareFontSeriesChangeRule {c}{el}{elc}{}%
40 \DeclareFontSeriesChangeRule {c}{l}{lc}{}%
41 \DeclareFontSeriesChangeRule {c}{sb}{sbc}{}%
42 \DeclareFontSeriesChangeRule {c}{sl}{slc}{}%
43 \DeclareFontSeriesChangeRule {c}{ub}{ubc}{}%
44 \DeclareFontSeriesChangeRule {c}{ul}{ulc}{}%
45 \DeclareFontSeriesChangeRule {c}{x}{x}{m}           %<-----

46 \DeclareFontSeriesChangeRule {ebc}{b}{bc}{}%
47 \DeclareFontSeriesChangeRule {ebc}{c}{ebc}{}%
48 \DeclareFontSeriesChangeRule {ebc}{eb}{ebc}{}%
49 \DeclareFontSeriesChangeRule {ebc}{ec}{ebec}{ebc}%
50 \DeclareFontSeriesChangeRule {ebc}{el}{elc}{}%
51 \DeclareFontSeriesChangeRule {ebc}{l}{lc}{}%
52 \DeclareFontSeriesChangeRule {ebc}{sb}{sbc}{}%
53 \DeclareFontSeriesChangeRule {ebc}{sc}{ebsc}{ebc}%
54 \DeclareFontSeriesChangeRule {ebc}{sl}{slc}{}%
55 \DeclareFontSeriesChangeRule {ebc}{ub}{ubc}{}%
56 \DeclareFontSeriesChangeRule {ebc}{ul}{ulc}{}%
57 \DeclareFontSeriesChangeRule {ebc}{x}{ebx}{}%

58 \DeclareFontSeriesChangeRule {ec}{bx} {bx} {b} %<-----
59 \DeclareFontSeriesChangeRule {ec}{b}{bec}{}%
60 \DeclareFontSeriesChangeRule {ec}{eb}{ebc}{}%
61 \DeclareFontSeriesChangeRule {ec}{el}{elec}{}%
62 \DeclareFontSeriesChangeRule {ec}{l}{lec}{}%
63 \DeclareFontSeriesChangeRule {ec}{sb}{sbec}{}%
64 \DeclareFontSeriesChangeRule {ec}{sl}{slec}{}%
65 \DeclareFontSeriesChangeRule {ec}{ub}{ubec}{}%
66 \DeclareFontSeriesChangeRule {ec}{ul}{ulec}{}%
67 \DeclareFontSeriesChangeRule {ec}{x}{x}{m}           %<-----

68 \DeclareFontSeriesChangeRule {sc}{bx} {bx} {b} %<-----
69 \DeclareFontSeriesChangeRule {sc}{b}{bsc}{}%

```

```

70 \DeclareFontSeriesChangeRule {sc}{eb}{ebsc}{}
71 \DeclareFontSeriesChangeRule {sc}{el}{elsc}{}
72 \DeclareFontSeriesChangeRule {sc}{l}{lsc}{}
73 \DeclareFontSeriesChangeRule {sc}{sb}{sbsc}{}
74 \DeclareFontSeriesChangeRule {sc}{s1}{slsc}{}
75 \DeclareFontSeriesChangeRule {sc}{ub}{ubsc}{}
76 \DeclareFontSeriesChangeRule {sc}{ul}{ulsc}{}
77 \DeclareFontSeriesChangeRule {sc}{x}{x}{m} %<-----

78 \DeclareFontSeriesChangeRule {ebx}{b}{bx}{}
79 \DeclareFontSeriesChangeRule {ebx}{c}{ebc}{}
80 \DeclareFontSeriesChangeRule {ebx}{eb}{ebx}{}
81 \DeclareFontSeriesChangeRule {ebx}{ec}{ebec}{}
82 \DeclareFontSeriesChangeRule {ebx}{el}{elx}{}
83 \DeclareFontSeriesChangeRule {ebx}{l}{lx}{}
84 \DeclareFontSeriesChangeRule {ebx}{sb}{sbx}{}
85 \DeclareFontSeriesChangeRule {ebx}{sc}{ebsc}{}
86 \DeclareFontSeriesChangeRule {ebx}{s1}{slx}{}
87 \DeclareFontSeriesChangeRule {ebx}{ub}{ubx}{}
88 \DeclareFontSeriesChangeRule {ebx}{ul}{ulx}{}
89 \DeclareFontSeriesChangeRule {ebx}{x}{ebx}{}

90 \DeclareFontSeriesChangeRule {eb}{c}{ebc}{}
91 \DeclareFontSeriesChangeRule {eb}{ec}{ebec}{}
92 \DeclareFontSeriesChangeRule {eb}{sc}{ebsc}{}
93 \DeclareFontSeriesChangeRule {eb}{x}{ebx}{}

94 \DeclareFontSeriesChangeRule {elc}{b}{bc}{}
95 \DeclareFontSeriesChangeRule {elc}{c}{elc}{}
96 \DeclareFontSeriesChangeRule {elc}{eb}{ebc}{}
97 \DeclareFontSeriesChangeRule {elc}{ec}{elec}{}
98 \DeclareFontSeriesChangeRule {elc}{el}{elc}{}
99 \DeclareFontSeriesChangeRule {elc}{l}{lc}{}
100 \DeclareFontSeriesChangeRule {elc}{sb}{sbc}{}
101 \DeclareFontSeriesChangeRule {elc}{sc}{elsc}{}
102 \DeclareFontSeriesChangeRule {elc}{s1}{slc}{}
103 \DeclareFontSeriesChangeRule {elc}{ub}{ubc}{}
104 \DeclareFontSeriesChangeRule {elc}{ul}{ulc}{}
105 \DeclareFontSeriesChangeRule {elc}{x}{elx}{}

106 \DeclareFontSeriesChangeRule {elx}{b}{bx}{}
107 \DeclareFontSeriesChangeRule {elx}{c}{elc}{}
108 \DeclareFontSeriesChangeRule {elx}{eb}{ebx}{}
109 \DeclareFontSeriesChangeRule {elx}{ec}{elec}{}
110 \DeclareFontSeriesChangeRule {elx}{el}{elx}{}
111 \DeclareFontSeriesChangeRule {elx}{l}{lx}{}
112 \DeclareFontSeriesChangeRule {elx}{sb}{sbx}{}
113 \DeclareFontSeriesChangeRule {elx}{sc}{elsc}{}
114 \DeclareFontSeriesChangeRule {elx}{s1}{slx}{}
115 \DeclareFontSeriesChangeRule {elx}{ub}{ubx}{}
116 \DeclareFontSeriesChangeRule {elx}{ul}{ulx}{}
117 \DeclareFontSeriesChangeRule {elx}{x}{elx}{}

118 \DeclareFontSeriesChangeRule {el}{c}{elc}{}
119 \DeclareFontSeriesChangeRule {el}{ec}{elec}{}
120 \DeclareFontSeriesChangeRule {el}{sc}{elsc}{}
121 \DeclareFontSeriesChangeRule {el}{x}{elx}{}

```

```

122 \DeclareFontSeriesChangeRule {lc}{b}{bc}={}
123 \DeclareFontSeriesChangeRule {lc}{c}{lc}={}
124 \DeclareFontSeriesChangeRule {lc}{eb}{ebc}={}
125 \DeclareFontSeriesChangeRule {lc}{ec}{lec}={}
126 \DeclareFontSeriesChangeRule {lc}{el}{elc}={}
127 \DeclareFontSeriesChangeRule {lc}{l}{lc}={}
128 \DeclareFontSeriesChangeRule {lc}{sb}{sbc}={}
129 \DeclareFontSeriesChangeRule {lc}{sc}{lsc}={}
130 \DeclareFontSeriesChangeRule {lc}{s1}{slc}={}
131 \DeclareFontSeriesChangeRule {lc}{ub}{ubc}={}
132 \DeclareFontSeriesChangeRule {lc}{ul}{ulc}={}
133 \DeclareFontSeriesChangeRule {lc}{x}{lx}={}

134 \DeclareFontSeriesChangeRule {lx}{b}{bx}={}
135 \DeclareFontSeriesChangeRule {lx}{c}{lc}={}
136 \DeclareFontSeriesChangeRule {lx}{eb}{ebx}={}
137 \DeclareFontSeriesChangeRule {lx}{ec}{lec}={}
138 \DeclareFontSeriesChangeRule {lx}{el}{elx}={}
139 \DeclareFontSeriesChangeRule {lx}{l}{lx}={}
140 \DeclareFontSeriesChangeRule {lx}{sb}{sbx}={}
141 \DeclareFontSeriesChangeRule {lx}{sc}{lsc}={}
142 \DeclareFontSeriesChangeRule {lx}{s1}{slx}={}
143 \DeclareFontSeriesChangeRule {lx}{ub}{ubx}={}
144 \DeclareFontSeriesChangeRule {lx}{ul}{ulx}={}
145 \DeclareFontSeriesChangeRule {lx}{x}{lx}={}

146 \DeclareFontSeriesChangeRule {l}{bx}{bx}{b} %<-----
147 \DeclareFontSeriesChangeRule {l}{b}{b}{bx} %<-----
148 \DeclareFontSeriesChangeRule {l}{c}{lc}{l} % ? %<-----
149 \DeclareFontSeriesChangeRule {l}{ec}{lec}{l} % ? %<-----
150 \DeclareFontSeriesChangeRule {l}{sb}{sb}{b} % ? %<-----
151 \DeclareFontSeriesChangeRule {l}{sc}{lsc}{l} % ? %<-----
152 \DeclareFontSeriesChangeRule {l}{x}{lx}{l} % ? %<-----

153 \DeclareFontSeriesChangeRule {m}{bx}{bx}{b} %<-----
154 \DeclareFontSeriesChangeRule {m}{b}{b}{bx} %<-----
155 \DeclareFontSeriesChangeRule {m}{c}{c}{m} %<-----
156 \DeclareFontSeriesChangeRule {m}{ec}{ec}{m} %<-----
157 \DeclareFontSeriesChangeRule {m}{l}{l}{m} %<-----
158 \DeclareFontSeriesChangeRule {m}{sb}{sb}{b} %<-----
159 \DeclareFontSeriesChangeRule {m}{sc}{sc}{m} %<-----
160 \DeclareFontSeriesChangeRule {m}{x}{x}{m} %<-----

161 \DeclareFontSeriesChangeRule {sbc}{b}{bc}={}
162 \DeclareFontSeriesChangeRule {sbc}{c}{sbc}={}
163 \DeclareFontSeriesChangeRule {sbc}{eb}{ebc}={}
164 \DeclareFontSeriesChangeRule {sbc}{ec}{sbec}{sbc}
165 \DeclareFontSeriesChangeRule {sbc}{el}{elc}={}
166 \DeclareFontSeriesChangeRule {sbc}{l}{lc}={}
167 \DeclareFontSeriesChangeRule {sbc}{sb}{sbc}={}
168 \DeclareFontSeriesChangeRule {sbc}{sc}{sbsc}{sbc}
169 \DeclareFontSeriesChangeRule {sbc}{s1}{slc}={}
170 \DeclareFontSeriesChangeRule {sbc}{ub}{ubc}={}
171 \DeclareFontSeriesChangeRule {sbc}{ul}{ulc}={}
172 \DeclareFontSeriesChangeRule {sbc}{x}{sbx}={}

173 \DeclareFontSeriesChangeRule {sbx}{b}{bx}={}

```

```

174 \DeclareFontSeriesChangeRule {sbx}{c}{sbc}{}  

175 \DeclareFontSeriesChangeRule {sbx}{eb}{ebx}{}  

176 \DeclareFontSeriesChangeRule {sbx}{ec}{sbec}{}  

177 \DeclareFontSeriesChangeRule {sbx}{el}{elx}{}  

178 \DeclareFontSeriesChangeRule {sbx}{l}{lx}{}  

179 \DeclareFontSeriesChangeRule {sbx}{sb}{sbx}{}  

180 \DeclareFontSeriesChangeRule {sbx}{sc}{sbsc}{}  

181 \DeclareFontSeriesChangeRule {sbx}{s1}{slx}{}  

182 \DeclareFontSeriesChangeRule {sbx}{ub}{ubx}{}  

183 \DeclareFontSeriesChangeRule {sbx}{ul}{ulx}{}  

184 \DeclareFontSeriesChangeRule {sbx}{x}{sbx}{}  

  

185 \DeclareFontSeriesChangeRule {sb}{c} {sbc} {bc} %? %<----  

186 \DeclareFontSeriesChangeRule {sb}{ec} {sbec} {sbc} %? %<----  

187 \DeclareFontSeriesChangeRule {sb}{sc} {sbsc} {sbc} %? %<----  

188 \DeclareFontSeriesChangeRule {sb}{x} {sbx} {bx} %? %<----  

  

189 \DeclareFontSeriesChangeRule {slc}{b}{bc}{}  

190 \DeclareFontSeriesChangeRule {slc}{c}{slc}{}  

191 \DeclareFontSeriesChangeRule {slc}{eb}{ebc}{}  

192 \DeclareFontSeriesChangeRule {slc}{ec}{slec}{}  

193 \DeclareFontSeriesChangeRule {slc}{el}{elc}{}  

194 \DeclareFontSeriesChangeRule {slc}{l}{lc}{}  

195 \DeclareFontSeriesChangeRule {slc}{sb}{sbc}{}  

196 \DeclareFontSeriesChangeRule {slc}{sc}{slsc}{}  

197 \DeclareFontSeriesChangeRule {slc}{s1}{slc}{}  

198 \DeclareFontSeriesChangeRule {slc}{ub}{ubc}{}  

199 \DeclareFontSeriesChangeRule {slc}{ul}{ulc}{}  

200 \DeclareFontSeriesChangeRule {slc}{x}{slx}{}  

  

201 \DeclareFontSeriesChangeRule {slx}{b}{bx}{}  

202 \DeclareFontSeriesChangeRule {slx}{c}{slc}{}  

203 \DeclareFontSeriesChangeRule {slx}{eb}{ebx}{}  

204 \DeclareFontSeriesChangeRule {slx}{ec}{slec}{}  

205 \DeclareFontSeriesChangeRule {slx}{el}{elx}{}  

206 \DeclareFontSeriesChangeRule {slx}{l}{lx}{}  

207 \DeclareFontSeriesChangeRule {slx}{sb}{sbc}{}  

208 \DeclareFontSeriesChangeRule {slx}{sc}{slsc}{}  

209 \DeclareFontSeriesChangeRule {slx}{s1}{slx}{}  

210 \DeclareFontSeriesChangeRule {slx}{ub}{ubx}{}  

211 \DeclareFontSeriesChangeRule {slx}{ul}{ulx}{}  

212 \DeclareFontSeriesChangeRule {slx}{x}{slx}{}  

  

213 \DeclareFontSeriesChangeRule {s1}{c}{slc}{}  

214 \DeclareFontSeriesChangeRule {s1}{ec}{slec}{}  

215 \DeclareFontSeriesChangeRule {s1}{sc}{slsc}{}  

216 \DeclareFontSeriesChangeRule {s1}{x}{slx}{}  

  

217 \DeclareFontSeriesChangeRule {ubc}{b}{bc}{}  

218 \DeclareFontSeriesChangeRule {ubc}{c}{ubc}{}  

219 \DeclareFontSeriesChangeRule {ubc}{eb}{ebc}{}  

220 \DeclareFontSeriesChangeRule {ubc}{ec}{ubec}{}  

221 \DeclareFontSeriesChangeRule {ubc}{el}{elc}{}  

222 \DeclareFontSeriesChangeRule {ubc}{l}{lc}{}  

223 \DeclareFontSeriesChangeRule {ubc}{sb}{sbc}{}  

224 \DeclareFontSeriesChangeRule {ubc}{sc}{ubsc}{}  

225 \DeclareFontSeriesChangeRule {ubc}{s1}{slc}{}  


```

```

226 \DeclareFontSeriesChangeRule {ubc}{ub}{ubc}{}
227 \DeclareFontSeriesChangeRule {ubc}{ul}{ulc}{}
228 \DeclareFontSeriesChangeRule {ubc}{x}{ubx}{}
229 \DeclareFontSeriesChangeRule {ubx}{b}{bx}{}
230 \DeclareFontSeriesChangeRule {ubx}{c}{ubc}{}
231 \DeclareFontSeriesChangeRule {ubx}{eb}{ebx}{}
232 \DeclareFontSeriesChangeRule {ubx}{ec}{ubec}{}
233 \DeclareFontSeriesChangeRule {ubx}{el}{elx}{}
234 \DeclareFontSeriesChangeRule {ubx}{l}{lx}{}
235 \DeclareFontSeriesChangeRule {ubx}{sb}{sbx}{}
236 \DeclareFontSeriesChangeRule {ubx}{sc}{ubsc}{}
237 \DeclareFontSeriesChangeRule {ubx}{s1}{slx}{}
238 \DeclareFontSeriesChangeRule {ubx}{ub}{ubx}{}
239 \DeclareFontSeriesChangeRule {ubx}{ul}{ulx}{}
240 \DeclareFontSeriesChangeRule {ubx}{x}{ubx}{}
241 \DeclareFontSeriesChangeRule {ub}{c}{ubc}{}
242 \DeclareFontSeriesChangeRule {ub}{ec}{ubec}{}
243 \DeclareFontSeriesChangeRule {ub}{sc}{ubsc}{}
244 \DeclareFontSeriesChangeRule {ub}{x}{ubx}{}
245 \DeclareFontSeriesChangeRule {ulc}{b}{bc}{}
246 \DeclareFontSeriesChangeRule {ulc}{c}{ulc}{}
247 \DeclareFontSeriesChangeRule {ulc}{eb}{ebc}{}
248 \DeclareFontSeriesChangeRule {ulc}{ec}{ulec}{ulc}
249 \DeclareFontSeriesChangeRule {ulc}{el}{elc}{}
250 \DeclareFontSeriesChangeRule {ulc}{l}{lc}{}
251 \DeclareFontSeriesChangeRule {ulc}{sb}{sbc}{}
252 \DeclareFontSeriesChangeRule {ulc}{sc}{ulsc}{ulc}
253 \DeclareFontSeriesChangeRule {ulc}{s1}{slc}{}
254 \DeclareFontSeriesChangeRule {ulc}{ub}{ubc}{}
255 \DeclareFontSeriesChangeRule {ulc}{ul}{ulc}{}
256 \DeclareFontSeriesChangeRule {ulc}{x}{ulx}{}
257 \DeclareFontSeriesChangeRule {ulx}{b}{bx}{}
258 \DeclareFontSeriesChangeRule {ulx}{c}{ulc}{}
259 \DeclareFontSeriesChangeRule {ulx}{eb}{ebx}{}
260 \DeclareFontSeriesChangeRule {ulx}{ec}{ulec}{}
261 \DeclareFontSeriesChangeRule {ulx}{el}{elx}{}
262 \DeclareFontSeriesChangeRule {ulx}{l}{lx}{}
263 \DeclareFontSeriesChangeRule {ulx}{sb}{sbx}{}
264 \DeclareFontSeriesChangeRule {ulx}{sc}{ulsc}{}
265 \DeclareFontSeriesChangeRule {ulx}{s1}{slx}{}
266 \DeclareFontSeriesChangeRule {ulx}{ub}{ubx}{}
267 \DeclareFontSeriesChangeRule {ulx}{ul}{ulx}{}
268 \DeclareFontSeriesChangeRule {ulx}{x}{ulx}{}
269 \DeclareFontSeriesChangeRule {ul}{c}{ulc}{}
270 \DeclareFontSeriesChangeRule {ul}{ec}{ulec}{}
271 \DeclareFontSeriesChangeRule {ul}{sc}{ulsc}{}
272 \DeclareFontSeriesChangeRule {ul}{x}{ulx}{}
273 \DeclareFontSeriesChangeRule {x}{b}{bx}{}
274 \DeclareFontSeriesChangeRule {x}{c}{c}{}
275 \DeclareFontSeriesChangeRule {x}{eb}{ebx}{}
276 \DeclareFontSeriesChangeRule {x}{ec}{ec}{}
277 \DeclareFontSeriesChangeRule {x}{el}{elx}{}

```

```

278 \DeclareFontSeriesChangeRule {x}{l}{lx}{}
279 \DeclareFontSeriesChangeRule {x}{sb}{sbx}{}
280 \DeclareFontSeriesChangeRule {x}{sc}{sc}{}
281 \DeclareFontSeriesChangeRule {x}{s1}{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 {s1}{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 {slec}{?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 of definition for \fontseries.*)

\delayed@f@adjustment The macro holding the delayed action(s) for use in `\selectfont`.

```

397 \let\delayed@f@adjustment\empty

```

(*End of 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 of 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 of 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 of 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 of 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 of 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 of 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 of 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 of 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 of 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 of 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 of 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 of 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 of 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 of 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 of 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 of 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 of 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 }  
665 }
```

(End of 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 of 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 of 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 of definition for \reinstall@nfss@defs.)

Supporting rollback ...

```
708 </2ekernel | latexrelease>
709 <latexrelease>\EndIncludeInRelease
710 <latexrelease>\IncludeInRelease{0000/00/00}%
711 <latexrelease>  {\merge@font@shape}{Font shape change rules}%
712 <latexrelease>
713 <latexrelease>\DeclareRobustCommand\fontshape [1]{\edef\f@shape{\#1}}
714 <latexrelease>\let\fontshapeforce@\undefined
715 <latexrelease>
716 <latexrelease>\let\merge@font@shape@\undefined
717 <latexrelease>\let\merge@font@shape@@\undefined
718 <latexrelease>
```

```

719 〈\latexrelease〉\let\merge@font@shape@without@substitution@undefined
720 〈\latexrelease〉\let\merge@font@shape@without@substitution@\@undefined
721 〈\latexrelease〉\let\delayed@merge@font@shape@undefined
722 〈\latexrelease〉
723 〈\latexrelease〉\let\normalshape@undefined
724 〈\latexrelease〉

```

This is always called in \document so don't make it undefined.

```

725 〈\latexrelease〉
726 〈\latexrelease〉\let\reinstall@nfss@defs\relax
727 〈\latexrelease〉\EndIncludeInRelease

```

This initializes the 2020/02/02 extensions to NFSS after any changes in the preamble.

```

728 〈*2ekernel | \latexrelease〉
729 〈\latexrelease〉\IncludeInRelease{2020/10/01}%
730 〈\latexrelease〉
731 〈\g@addto@macro\@kernel@after@begindocument@before
732 〈\reinstall@nfss@defs\init@series@setup〉
733 〈/2ekernel | \latexrelease〉
734 〈\latexrelease〉\EndIncludeInRelease

```

The initialization was introduced in 2020/02/02 but

```

735 〈\latexrelease〉\IncludeInRelease{2020/02/02}%
736 〈\latexrelease〉
737 〈\latexrelease〉\AtBeginDocument{\reinstall@nfss@defs\init@series@setup}
738 〈\latexrelease〉\EndIncludeInRelease
739 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
740 〈\latexrelease〉
741 〈\latexrelease〉\EndIncludeInRelease
742 〈*2ekernel〉
743 〈/2ekernel〉

```

File x

ltfsstrc.dtx

1 Introduction

This package contains the code for tracing font loading and font changes. It basically overlays some of the low-level functions of NFSS with additional code used for tracing.

The package accepts the following options:

errorshow Write all information about font changes etc. only to the transcript file unless an error happens. This means that information about font substitution will not be shown on the terminal.

warningshow Show all NFSS warnings on the terminal. This setting corresponds to the default behaviour of NFSS if the **tracefnt** package is *not* loaded!

infoshow Show all NFSS warning and all NFSS info messages (that are normally only written to the transcript file) also on the terminal. This is the default if the **tracefnt** package is loaded.

debugshow In addition to **infoshow** show also changing of math fonts as far as possible (this option can produce a large amount of output).

loading Show the name of external fonts when they are loaded. This option shows only “newly” loaded fonts not those already preloaded in the format or the class file before the **tracefnt** package became active.

pausing Turn all font warnings into errors so that L^AT_EX will stop.

2 A driver for this document

The next bit of code contains the documentation driver file for T_EX, 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^AT_EX 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`.²⁹

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 of 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]}}%
```

²⁹This 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`³⁰ 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 of 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
```

³⁰This 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 of 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^AT_EX's `\twfbf`, etc. We define `\font@name` globally, as always. The reason for this is explained later on.

```
141      \xdef\font@name{%
142          \csname\curr@fontshape/\f@size\endcsname}%
```

We call the macro `\pickup@font` which will load the font if necessary.

```
143      \pickup@font
```

Then we select the font.

```
144      \font@name
```

After switching fonts we run a hook, so that packages can make last minute alterations based on the new font (originally provided in `everysel` but using a different interface).

```
145      \UseHook{selectfont}%
```

Finally we call `\size@update`. This macro is normally empty but will contain actions (like setting the `\baselineskip`) that have to be carried out when the font size, the base `\baselineskip` or the `\baselinestretch` have changed.

```
146      \size@update
```

A similar function is called to handle anything related to encoding updates. This one is changed from `\relax` by `\fontencoding`.

```
147     \enc@update
```

Just before ending this macro we have to pop the tracing stack if it was pushed before.

```
148 <+debug> \poptracing
149 }
```

(End of definition for `\selectfont`.)

selectfont Declare the hook used in `selecfont` in the kernel, but not inside the `tracefnt` package.

```
150 <-trace> \NewHook{selectfont}
```

(End of 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   \Qdefaultunits\Qtempdimb#2pt\relax\Qnil
```

```

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  {*trace}
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 of 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 of 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 of 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 of 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 of 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 of 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 of 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 of 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 of 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_EX* macros for placing accents.

(End of 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 of 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.³¹ 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 of 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 it argument.

```

343 
```

```

344 \def\ifnot@nil#1{\def\reserved@a{#1}%
345   \ifx\reserved@a\@nil \expandafter\@gobble
346   \else \expandafter\@firstofone\fi}

```

(End of 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}

```

³¹One 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 of 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\@nil\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 of 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 of 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 of 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 of 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@single\else
379          \expandafter\check@range\fi}

```

(End of 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\@nil{%
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\@nil}%

```

We have to make sure that both boundaries are present, if not we have to set them. Here we check the upper bound. If `\upper@bound` is zero after the assignment we set it to `\maxdimen` (upper open range). We need to use a $\langle dimen \rangle$ register for the scan since we may have a decimal number as the boundary.

```
383     \upper@bound0#2\p@
384     \ifdim\upper@bound=\z@ \upper@bound\maxdimen\fi
```

Now we check the upper boundary against `\f@size`. If it is larger or equal than `\f@size` this range is no good and we have to recurse.

```
385     \ifdim \f@size \p@<\upper@bound
```

Otherwise we have to check the lower bound. This time it is not necessary to scan the boundary value into a register because if it is empty we get zero as desired. We could even omit the 0 which would result in 1pt as default lower boundary. If `\f@size` is smaller than the boundary we have to recurse.

```
386     \lower@bound0#1\p@
387     \ifdim \f@size \p@<\lower@bound
388     \else
```

If both tests are passed we can try executing the size function.

```
389     \set@simple@size@args#3<#4\@nnil
390     \execute@size@function\sizefn@info
```

If the function was successful it should have left an external font name in `\external@font`. We use this to see if we can stop scanning. Otherwise we recurse.

```
391     \ifx\external@font\@empty
392     \else
393         \let\reserved@f\@empty
394         \fi
395     \fi
396     \fi
397     \reserved@f}}
```

(End of 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 of 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 of definition for `\check@single`.)

`\set@size@funct@args` This macro sets the optional and mandatory arguments for a size function. If the optional argument is not present it is set to the empty token list. The mandatory argument is delimited by the token `\@nil`.

```
411 \def\set@size@funct@args{\@ifnextchar[%]
412   \set@size@funct@args{\set@size@funct@args[]}}
413 \def\set@size@funct@args[#1]{\@nil{%
414   \def\mandatory@arg{#1}%
415   \def\optional@arg{}%
416 }}
```

(End of definition for `\set@size@funct@args` and `\set@size@funct@args@`.)

`\DeclareSizeFunction` This function defines a new size function hiding the internal from the designer. The body of the size function may use `\optional@arg` and `\mandatory@arg` denoting the optional and mandatory argument that may follow the size specification `<...>`.

```
417 {*2ekernel}
418 \def\DeclareSizeFunction#1#2{\@namedef{s@fct@#1}{#2}}
419 \@onlypreamble\DeclareSizeFunction
420 
```

(End of definition for `\DeclareSizeFunction`.)

`\execute@size@function` This macro is very simple. The only point worth noting is that calling an undefined size function will do nothing (actually execute a `\relax`).

```
421 {*2ekernel | package}
422 \def\execute@size@function#1{%
423   {*trace}
424   \@ifundefined{s@fct@#1}{%
425     {\errmessage{Undefined font size function #1}%
426      \s@fct@}%
427     {\csname s@fct@#1\endcsname}%
428   }%
429   {-trace} \csname s@fct@#1\endcsname
430 }
431 
```

(End of 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 of 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 of 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 of 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 of 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 of 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 of definition for `\s@fct@`.)

- `\s@fct@gens` 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 of definition for `\s@fct@gens` 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 of 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 of 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 of 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 of 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}{\subf@sfcnt@\font@warning}
558 \DeclareSizeFunction{ssubf}{\subf@sfcnt@\font@info}
559 \def\subf@sfcnt#1{%
560   #1{Font\space shape\space ‘\curr@fontshape’\space in\space
561   size\space f@size\space not\space available\MessageBreak
562   external\space font\space ‘\mandatory@arg’\space used}%
563 \empty@sfcnt#1%
564 }
```

(End of 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 of definition for *\s@fct@fixed*.)

File y

ltfsscmp.dtx

This file contains the implementation of commands giving compatibility with the original ‘NFSS1’ release of the Font Selection Scheme.

Warning: The macro documentation is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

Version 1 of NFSS is obsolete now for about 20 years (and was “current” only for a short intermediate time) so with the 2015 release these internal interface commands are removed from the kernel and made available via `latexrelease` package so that backward compatibility remains ensured for very old documents.

```
1  {*latexrelease}
2  \IncludeInRelease{2015/01/01}{\new@fontshape}%
3  {NFSS version1 commands}%
4  \let\new@fontshape@\undefined
5  \let\warn@rel@i@\undefined
6  \let\scan@fontshape@\undefined
7  \let\scan@@fontshape@\undefined
8  \let\subst@fontshape@\undefined
9  \let\extra@def@\undefined
10 \let\default@mextra@\undefined
11 \let\preload@sizes@\undefined
12 \let\err@rel@i@\undefined
13 \let\newmathalphabet@\undefined
14 \let\newmathalphabet@@\undefined
15 \let\newmathalphabet@@@\undefined
16 \let\if@no@font@opt@\undefined
17 \let\@no@font@opt@false@\undefined
18 \let\define@mathalphabet@\undefined
19 \let\define@mathgroup@\undefined
20 \let\addtoversion@\undefined
21 \EndIncludeInRelease
```

In older releases we provide the original definitions.

```
22 \IncludeInRelease{0000/00/00}{\new@fontshape}%
23 {NFSS version1 commands}%
```

`\new@fontshape` The interface is now `\DeclareFontShape`.

```
24 \gdef\new@fontshape#1#2#3#4{%
25   \warn@rel@i\new@fontshape\DeclareFontShape
26   \expandafter\scan@fontshape\@gobble#4<\@nil><<%
27   \DeclareFontShape U{#1}{#2}{#3}\reserved@f}%
28 \onlypreamble\new@fontshape
```

(End of 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 of 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 of 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<#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 of 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 of 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 of 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 of 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 of definition for `\preload@sizes`.)

`\err@rel@i` This macro is used in cases where emulation with NFSS2 features is not really possible.

```

99 \gdef\err@rel@i#1#2{%
100   \o@late@error{*** NFSS release 1 command \noexpand#1found%
101     ^^J*** Recovery not possible. Use \string#2}%
102   {The new release of NFSS doesn't support the
103     \noexpand#1command^^Jany longer.
104     Please upgrade your file to the syntax of NFSS
105     release 2^^Jusing the \noexpand#2command.}%

```

Let's die.

```
106   \batchmode\input.\relax
107 }%
108 \onlypreamble\err@rel@i
```

(End of 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 of 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 of 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 of 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 of 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 of definition for \addtoversion.)  
Finishing off this huge \IncludeInRelease argument:  
143 \EndIncludeInRelease  
144 </latexrelease>
```

File z

ltfssdcl.dtx

This file contains the main implementation of the font selection scheme commands. See other parts of the L^AT_EX distribution, or *The L^AT_EX 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@ \cin 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 of 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><font info>
...
\@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 of definition for $\backslash version@list$.)

\version@elt

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

(End of 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 of 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 of definition for \alpha@list.)

`\alpha@elt`

```
45 \let\alpha@elt\relax
46 \onlypreamble\alpha@elt
```

(End of definition for \alpha@elt.)

`\newgroup` Start the group (fam) allocation at 0. (Doesn't belong here.)

```
47 \count18=-1
```

(End of definition for \newgroup.)

`\stepcounter`

(End of 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  \if@eha
100 \fi
101 \else \expandafter\non@alpherr\fi
102 #1{#4}%
103 }%
104 }
105 \EndIncludeInRelease
106 {*2ekernel}
107 \onlypreamble\restore@mathversion

```

(End of 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 of 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 of 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 of definition for \dorestore@version.)

`\c@localmathalphabets` 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 of definition for `\c@localmathalphabets`.)

`\document@select@group` The `\document@select@group` command is the version of `\select@group` (inside math versions) that is used in the document body to set up math alphabets (if used).

```

145  \def\document@select@group#1#2#3#4{%
146  \ifx\math@bgroup\math@bgroup\else\relax\expandafter\@firstofone\fi
147  {%
148  \ifmmode

```

We first check if there is still room for allocating another mathgroup. If there is, we check if it can be globally allocated or if we have reached the limit which is given by `\e@mathgroup@top` with `\c@localmathalphabets` subtracted.

```

149  \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
150  \ifnum \numexpr\mathgroup@top-\c@localmathalphabets
151  >\csname c@mv@\math@version\endcsname
152  \else

```

If we are past this point we freeze the current state of the math version so that we can return to it after the formula has ended. Of course, that should be done only once, so we check if `\mv@<version>@frozen` already exists.

```
153  \ifcsname mv@\math@version @frozen\endcsname \else
```

We have to pass the current value of `\math@version` not the macro itself, because some of the processing is delayed to a point where the value may have changed again—not doing this caused a puzzling error in one setup.

```

154  \expandafter\freeze@math@version\expandafter{\math@version}%
155  \fi
156  \fi
157  \begingroup
158  \escapechar\m@ne
159  \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
160  \globaldefs\one \math@fonts

```

```

161      \endgroup
162      \expandafter\extract@alph@from@version
163          \csname mv@\math@version\expandafter\endcsname
164          \expandafter{\number\csname
165              c@mv@\math@version\endcsname}%
166          #1%
167          \global\advance\csname c@mv@\math@version\endcsname\@ne
168      \else
169          \let#1\relax
170          \@latex@error{Too many math alphabets used in
171              version \math@version}%
172          \@eha
173      \fi

```

Extra \expandafter to remove the \expandafter added below

```
174      \else \expandafter\expandafter\expandafter\non@alpherr\fi
```

We surround \select@group with braces so that functions using it can be used directly after _ or ^.

If the legacy interface is used, e.g., \$ $\$sf -1$$ the math alphabet #1 does not take an argument so we better do not surround #4 with braces, because then we get {\relax} into the formula and introduce an extra Ord atom. The two different cases can be distinguished by looking at the current value of \math@bgroup.

```

175      \expandafter#1\ifx\math@bgroup\bgroup{\#4}\else#4\fi
176      }%
177  }
178  </2ekernel | latexrelease>
179  <| latexrelease>\EndIncludeInRelease
180  <| latexrelease>\IncludeInRelease{2020/10/01}
181  <| latexrelease>  {\document@select@group}{\document@select@group}%
182  <| latexrelease>
183  <| latexrelease>\def\document@select@group#1#2#3#4{%
184  <| latexrelease> \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
185  <| latexrelease> {%
186  <| latexrelease> \ifmmode
187  <| latexrelease>   \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
188  <| latexrelease>     \begingroup
189  <| latexrelease>       \escapechar\m@ne
190  <| latexrelease>       \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
191  <| latexrelease>       \globaldefs\@ne \math@fonts
192  <| latexrelease>     \endgroup
193  <| latexrelease>   \expandafter\extract@alph@from@version
194  <| latexrelease>       \csname mv@\math@version\expandafter\endcsname
195  <| latexrelease>       \expandafter{\number\csname
196  <| latexrelease>           c@mv@\math@version\endcsname}%
197  <| latexrelease>           #1%
198  <| latexrelease>       \global\advance\csname c@mv@\math@version\endcsname\@ne
199  <| latexrelease>   \else
200  <| latexrelease>     \let#1\relax
201  <| latexrelease>     \@latex@error{Too many math alphabets used
202  <| latexrelease>                     in version \math@version}%
203  <| latexrelease>     \@eha
204  <| latexrelease>   \fi
205  <| latexrelease> \else \expandafter\expandafter\expandafter\non@alpherr\fi

```

```

206 〈\latexrelease〉 \expandafter#1\ifx\math@bgroup\bgroup{#4}\else#4\fi
207 〈\latexrelease〉 }%
208 〈\latexrelease〉
209 〈\latexrelease〉\EndIncludeInRelease
210 〈\latexrelease〉\IncludeInRelease{2015/01/01}
211 〈\latexrelease〉 {\document@select@group}{\document@select@group}%
212 〈\latexrelease〉
213 〈\latexrelease〉\def\document@select@group#1#2#3#4{%
214 〈\latexrelease〉 \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
215 〈\latexrelease〉 {%
216 〈\latexrelease〉 \ifmmode
217 〈\latexrelease〉 \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
218 〈\latexrelease〉 \begin{group}
219 〈\latexrelease〉 \escapechar\m@ne
220 〈\latexrelease〉 \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
221 〈\latexrelease〉 \globaldefs\@ne \math@fonts
222 〈\latexrelease〉 \end{group}
223 〈\latexrelease〉 \expandafter\extract@alph@from@version
224 〈\latexrelease〉 \csname mv@\math@version\expandafter\endcsname
225 〈\latexrelease〉 \expandafter{\number\csname
226 〈\latexrelease〉 c@mv@\math@version\endcsname}%
227 〈\latexrelease〉 #1%
228 〈\latexrelease〉 \global\advance\csname c@mv@\math@version\endcsname\@ne
229 〈\latexrelease〉 \else
230 〈\latexrelease〉 \let#1\relax
231 〈\latexrelease〉 \@latex@error{Too many math alphabets used
232 〈\latexrelease〉 in version \math@version}%
233 〈\latexrelease〉 \relax
234 〈\latexrelease〉 \fi
235 〈\latexrelease〉 \else \expandafter\non@alpherr\fi
236 〈\latexrelease〉 #1{#4}%
237 〈\latexrelease〉 }%
238 〈\latexrelease〉
239 〈\latexrelease〉\EndIncludeInRelease
240 〈\latexrelease〉
241 〈\latexrelease〉\IncludeInRelease{0000/00/00}
242 〈\latexrelease〉 {\document@select@group}{\document@select@group}%
243 〈\latexrelease〉
244 〈\latexrelease〉\def\document@select@group#1#2#3#4{%
245 〈\latexrelease〉 \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
246 〈\latexrelease〉 {%
247 〈\latexrelease〉 \ifmmode
248 〈\latexrelease〉 \ifnum\csname c@mv@\math@version\endcsname<\sixt@n
249 〈\latexrelease〉 \begin{group}
250 〈\latexrelease〉 \escapechar\m@ne
251 〈\latexrelease〉 \getanddefine@fonts
252 〈\latexrelease〉 {\csname c@mv@\math@version\endcsname}#3%
253 〈\latexrelease〉 \globaldefs\@ne \math@fonts
254 〈\latexrelease〉 \end{group}
255 〈\latexrelease〉 \expandafter\extract@alph@from@version
256 〈\latexrelease〉 \csname mv@\math@version\expandafter\endcsname
257 〈\latexrelease〉 \expandafter{\number\csname
258 〈\latexrelease〉 c@mv@\math@version\endcsname}%
259 〈\latexrelease〉 #1%

```

```

260 <|latexrelease>      \global\advance\csname c@mv@\math@version\endcsname\@ne
261 <|latexrelease>      \else
262 <|latexrelease>      \let#1\relax
263 <|latexrelease>      \@latex@error{Too many math alphabets used
264 <|latexrelease>          in version \math@version}%
265 <|latexrelease>      \@eha
266 <|latexrelease>      \fi
267 <|latexrelease>      \else \expandafter\non@alpherr\fi
268 <|latexrelease>      #1{#4}%
269 <|latexrelease>  }%
270 <|latexrelease>}
271 <|latexrelease>\EndIncludeInRelease
272 <|2ekernel>

```

(End of definition for `\document@select@group.`)

`\freeze@math@version` 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.

```

273 <|2ekernel>
274 <|2ekernel | latexrelease>
275 <|latexrelease>\IncludeInRelease{2022/11/01}%
276 <|latexrelease>          {\freeze@math@version}{freeze math version}%
277 \ExplSyntaxOn
278 \cs_new_protected:Npn\freeze@math@version #1 {

```

Save the current `\mv@<version>` code and the number of allocated mathgroups inside.

```

279   @font@info{Freeze~ math~ alphabet~ allocation~ in~ version-
280           #1.\MessageBreak
281           Allocated~math~groups:~\int_use:c{ c@mv@ #1 }~
282           (local:~ \int_use:N\c@localmathalphabets)      }
283 \cs_gset_eq:cc { mv@#1@frozen }{ mv@#1 }
284 \tl_gset:cx { g__nfss_frozen_mv_ #1 _tl }{ \int_use:c { c@mv@#1 } }

```

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.

```

285 \cs_gset:cpn{mv@#1@reset}
286   {
287     \int_compare:nNnTF { \int_use:c{c@mv@#1} } >
288       { \tl_use:c{g__nfss_frozen_mv_ #1 _tl} }
289   {
290     @font@info{Undo~ math~ alphabet~ allocation~ in~ version~ #1}

```

If the undo is necessary, we restore the `\mv@<version>` code.

```

291   \cs_gset_eq:cc { mv@#1 }{ mv@#1@frozen }
292   \int_gset:cn { c@mv@#1 }{ \tl_use:c {g__nfss_frozen_mv_ #1 _tl} }

```

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.

```

293   \group_begin:
294     \cs_set_eq:NN \getanddefine@fonts \use_none:nn
295     \use:c {mv@#1}
296     \group_end:
297   }
298   {

```

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

```
299     \@font@info{No~ math~ alphabet~ change~ to~ frozen~ version~ #1}
300 }
```

If this is executed after a math display, we may have to arrange for ignoring spaces, because they are now hidden if the tokens from above intervene. This is signaled by the 2e switch `@ignore` which is set in `\frozen@everymath` and `\frozen@everydisplay`.

This is all 2e code so we use that syntax.

```
301     \if@ignore \ignorespaces \fi
302 }
303 }
304 \ExplSyntaxOff
305 </2ekernel | latexrelease>
306 <latexrelease>\EndIncludeInRelease
307 <latexrelease>\IncludeInRelease{2021/11/15}
308 <latexrelease>           {\freeze@math@version}{freeze math version}%
309 <latexrelease>
310 <latexrelease>\ExplSyntaxOn
311 <latexrelease>\cs_set_protected:Npn\freeze@math@version #1 {
312 <latexrelease>   \@font@info{Freeze~ math~ alphabet~ allocation~ in~ version~ #1.\MessageBreak
313 <latexrelease>           Allocated~math~groups:~\int_use:c{ c@mv@ #1 }~%
314 <latexrelease>           (local:~ \int_use:N\c@localmathalphabets)      }
315 <latexrelease>
316 <latexrelease> \cs_gset_eq:cc { mv@#1@frozen }{ mv@#1 }
317 <latexrelease> \tl_gset:cx { g__nfss_frozen_mv_ #1 _tl }{ \int_use:c { c@mv@#1 } }
318 <latexrelease> \group_insert_after:N \__nfss_init_mv_freeze:N
319 <latexrelease> \exp_after:wN \group_insert_after:N \cs:w mv@#1@reset \cs_end:
320 <latexrelease> \tl_gput_right:No \check@mathfonts
321 <latexrelease>   {
322 <latexrelease>     \exp_after:wN \group_insert_after:N \cs:w mv@#1@reset \cs_end:
323 <latexrelease>   }
324 <latexrelease> \cs_gset:cpn{mv@#1@reset}
325 <latexrelease>   {
326 <latexrelease>     \int_compare:nNnTF { \int_use:c{c@mv@#1} } >
327 <latexrelease>       { \tl_use:c{g__nfss_frozen_mv_ #1 _tl} }
328 <latexrelease>   {
329 <latexrelease>     \@font@info{Undo~ math~ alphabet~ allocation~ in~ version~ #1}
330 <latexrelease>     \cs_gset_eq:cc { mv@#1 }{ mv@#1@frozen }
331 <latexrelease>     \int_gset:cn { c@mv@#1 }{ \tl_use:c { g__nfss_frozen_mv_ #1 _tl} }
332 <latexrelease>     \group_begin:
333 <latexrelease>       \cs_set_eq:NN \getanddefine@fonts \use_none:nn
334 <latexrelease>       \use:c { mv@#1 }
335 <latexrelease>     \group_end:
336 <latexrelease>   }
337 <latexrelease>   {
338 <latexrelease>     \@font@info{No~ math~ alphabet~ change~ to~ frozen~ version~ #1}
339 <latexrelease>   }
340 <latexrelease> \if@ignore \ignorespaces \fi
341 <latexrelease> }
342 <latexrelease>
343 <latexrelease>\cs_set_protected:Npn \__nfss_init_mv_freeze:N #1 {%
344 <latexrelease>   \mode_if_math:T { \group_insert_after:N \__nfss_init_mv_freeze:N
345 <latexrelease>           \group_insert_after:N } #1
346 <latexrelease>}
```

```

347  ⟨latexrelease⟩\ExplSyntaxOff
348  ⟨latexrelease⟩
349  ⟨latexrelease⟩\EndIncludeInRelease
350  ⟨*2ekernel⟩

```

(End of definition for \freeze@math@version.)

\process@table

```

351  \def\process@table{%
352      \def\cdp@elt##1##2##3##4{%
353          \o@font@info{Checking defaults for
354              ##1##2##3##4}%
355          \expandafter
356          \ifx\csname##1##2##3##4\endcsname\relax
357              \begingroup
358                  \def\f@encoding{##1}\def\f@family{##2}%
359                  \try@load@fontshape
360              \endgroup
361          \fi
362          \expandafter
363          \ifx\csname##1##2##3##4\endcsname\relax
364              \@latex@error{This NFSS system isn't set up properly}%
365                  {For encoding scheme ##1 the defaults
366                      ##2##3##4 do not form a valid font shape}%
367          \else
368              \o@font@info{... okay}%
369          \fi}%
370      \cdp@list

```

Now we make sure that \error@fontshape is okay.

```

371  \begingroup
372      \escapechar\m@ne
373      \error@fontshape
374      \expandafter\ifx\csname \curr@fontshape\endcsname\relax
375          \begingroup
376              \try@load@fontshape
377          \endgroup
378      \fi
379      \expandafter\ifx\csname \curr@fontshape\endcsname\relax
380          \@latex@error{This NFSS system isn't set up properly}%
381              {The system maintainer forgot to specify a suitable
382                  substitution
383                  font shape using the \noexpand\DeclareErrorFont
384                  command}%
385      \fi
386  \endgroup

```

Set \select@group to its meaning used within the document body.

```

387  \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`.

```

388      \fontencoding\encodingdefault
389      \edef\f@family{\familydefault}%
390      \edef\f@series{\seriesdefault}%
391      \edef\f@shape{\shapedefault}%

```

Drop stuff not longer needed. We need to add many more!!!!!!

```

392  \everyjob{}%
393 }
394 \onlypreamble\process@table

```

(End of definition for `\process@table`.)

```

395 \%onlypreamble\set@mathradical

```

`\DeclareMathVersion`

```

396 </2ekernel>
397 <*2ekernel | latexrelease>
398 <latexrelease>\IncludeInRelease{2022/11/01}%
399 <latexrelease>      {\DeclareMathVersion}{local alphabets}%
400 \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.

```
401  \@namedef{g_nfss_frozen_mv_#1_t1}{}%
```

We also extend `\check@mathfonts` to call a version reset (once frozen) after a formula has finished.

```

402  \expandafter\ifx\csname mv@#1\endcsname \relax
403    \expandafter \g@addto@macro \expandafter \check@mathfonts
404    \expandafter {\expandafter \aftergroup \csname mv@#1@reset\endcsname}%

```

Initially this macro does nothing. It is, however, important that it doesn't stop any `\ignorespaces`, so we make it expandable and not `\relax`.

```

405    \@namedef{mv@#1@reset}{}%
406  \fi

```

```

407  \expandafter\new@mathversion\csname mv@#1\endcsname
408 \onlypreamble\DeclareMathVersion
409 </2ekernel | latexrelease>
410 <latexrelease>\EndIncludeInRelease

```

```

411 <latexrelease>\IncludeInRelease{2021/11/15}%
412 <latexrelease>      {\DeclareMathVersion}{local alphabets}%
413 <latexrelease>\def\DeclareMathVersion#1{%
414 <latexrelease>  \@namedef{g_nfss_frozen_mv_#1_t1}{}%
415 <latexrelease>  \expandafter\new@mathversion\csname mv@#1\endcsname}
416 <latexrelease>\EndIncludeInRelease

```

```

417 <latexrelease>\IncludeInRelease{0000/00/00}%
418 <latexrelease>      {\DeclareMathVersion}{local alphabets}%
419 <latexrelease>\def\DeclareMathVersion#1{%
420 <latexrelease>  \expandafter\new@mathversion\csname mv@#1\endcsname}
421 <latexrelease>\EndIncludeInRelease

```

```

422 <*2ekernel>

```

(End of definition for \DeclareMathVersion.)

```
\new@mathversion
```

```
423 \def\new@mathversion#1{%
424   \expandafter\in@\expandafter#\expandafter{\version@list}%
425   \ifin@
426     \@font@info{Redeclaring math version
427       '\expandafter\gobblefour\string#1'}%
428   \else
429     \expandafter\newcount\csname c@\expandafter
430           \gobble\string#1\endcsname
431     \def\version@elt{\noexpand\version@elt\noexpand}%
432     \edef\version@list{\version@list\version@elt#1}%
433   \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.

```
434   \toks@{}%
435   \count@z@
```

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.

```
436 \def\group@elt##1##2{%
437   \advance\count@\@ne
438   \addto@hook\toks@{\getanddefine@fonts##1##2}%
439   }%
440 \group@list
```

We set the counter for this math version to the number of math groups found in \group@list.

```
441 \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.

```
442 \def\alpha@elt##1##2##3{%
443   \ifx##2\no@alphabet@error
444     \toks@{\expandafter{\the\toks@\install@mathalphabet##1%
445     {\no@alphabet@error##1}}}
446   \else
447     \toks@{\expandafter{\the\toks@\install@mathalphabet##1%
448     {\select@group##1##2##3}}}
449   \fi
450   }%
451 \alpha@list
```

Finally we define the math version to expand to the contents of \toks@.

```
452 \xdef#1{\the\toks@}%
453 }
454 \onlypreamble\new@mathversion
```

(End of definition for \new@mathversion.)

\DeclareSymbolFont First drop any surplus m from the series argument then do what has been done since 1994.

```
455 </2ekernel>
456 <*2ekernel | latexrelease>
```

```

457 〈\latexrelease〉\IncludeInRelease{2022/11/01}%
458 〈\latexrelease〉                                {\DeclareSymbolFont}{maybe drop m}%
459 \def\DeclareSymbolFont #1#2#3#4#5{%
460   \def\reserved@a{\DeclareSymbolFont@m@dropped{#1}{#2}{#3}}%
461   \edef\reserved@b{#4}%
462   \series@maybe@drop@one@m\reserved@b\reserved@b
463   \expandafter\reserved@a\expandafter{\reserved@b}{#5}%
464 }
465 \def\DeclareSymbolFont@m@dropped #1#2#3#4#5{%
466   \tempswafalse
467   \edef\reserved@b{#2}%
468   \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
469     \ifx\reserved@b\reserved@c \tempswatrue\fi}%
470   \cdp@list
471   \if@tempswa
472     \ifundefined{sym#1}%
473       \ifnum\count18<15 %
474         \expandafter\new@mathgroup\csname sym#1\endcsname
475         \expandafter\new@symbolfont\csname sym#1\endcsname
476           {#2}{#3}{#4}{#5}%
477       \else
478         \@latex@error{Too many symbol fonts declared}\@eha
479       \fi
480     }%
481   }%
482   \font@info{Redeclaring symbol font '#1'}%

```

Update the group list.

```

483 \def\group@elt##1##2{%
484   \noexpand\group@elt\noexpand##1%
485   \expandafter\ifx\csname sym#1\endcsname##1%
486     \expandafter\noexpand\csname##2/#3/#4/#5\endcsname
487   \else
488     \noexpand##2%
489   \fi}%
490 \xdef\group@list{\group@list}%

```

Update the version list.

```

491 \def\version@elt##1{%
492   \expandafter
493   \SetSymbolFont@\expandafter##1\csname##2/#3/#4/#5\expandafter
494     \endcsname \csname sym#1\endcsname
495   }%
496   \version@list
497 }%
498 \else
499   \@latex@error{Encoding scheme '#2' unknown}\@eha
500 \fi
501 }%
502 \onlypreamble\DeclareSymbolFont
503 (/2ekernel | \latexrelease)
504 〈\latexrelease〉\EndIncludeInRelease
505 〈\latexrelease〉\IncludeInRelease{0000/00/00}%

```

```

506  ⟨latexrelease⟩          {\DeclareSymbolFont}{maybe drop m}%
507  ⟨latexrelease⟩
508  ⟨latexrelease⟩\let\DeclareSymbolFont\DeclareSymbolFont@m@dropped
509  ⟨latexrelease⟩\let\DeclareSymbolFont@m@dropped\@undefined
510  ⟨latexrelease⟩
511  ⟨latexrelease⟩\EndIncludeInRelease
512  {*2ekernel}

(End of definition for \DeclareSymbolFont.)
```

\group@list

```

513  \let\group@list\empty
514  \@onlypreamble\group@list

(End of definition for \group@list.)
```

\group@elt

```

515  \let\group@elt\relax
516  \@onlypreamble\group@elt

(End of definition for \group@elt.)
```

\new@symbolfont

```

517  \def\new@symbolfont#1#2#3#4#5{%
518    \toks@\expandafter{\group@list}%
519    \edef\group@list{\the\toks@\noexpand\group@elt\noexpand#1%
520      \expandafter\noexpand\csname#2/#3/#4/#5\endcsname}%
521    \def\version@elt##1{\toks@\expandafter{##1}%
522      \edef##1{\the\toks@\noexpand\getanddefine@fonts
523        #1\expandafter\noexpand\csname#2/#3/#4/#5\endcsname}%
524      \global\advance\csname c@\expandafter
525        \@gobble\string##1\endcsname\@ne
526    }%
527    \version@list
528  }
529  \@onlypreamble\new@symbolfont

(End of definition for \new@symbolfont.)
```

\SetSymbolFont First drop any surplus m from the series argument then do what has been done since 1994.

```

530  ⟨/2ekernel⟩
531  ⟨*2ekernel | latexrelease⟩
532  ⟨latexrelease⟩\IncludeInRelease{2022/11/01}%
533  ⟨latexrelease⟩          {\SetSymbolFont}{maybe drop m}%
534  \def\SetSymbolFont #1#2#3#4#5#6{%
535    \def\reserved@a{\SetSymbolFont@m@dropped{#1}{#2}{#3}{#4}}%
536    \edef\reserved@b{#5}%
537    \series@maybe@drop@one@m\reserved@b\reserved@b
538    \expandafter\reserved@a\expandafter{\reserved@b}{#6}%
539  }
```

```

540 \def\SetSymbolFont@m@dropped#1#2#3#4#5#6{%
541   \tempswafalse
542   \edef\reserved@b{#3}%
543   \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
544     \ifx\reserved@b\reserved@c \tempswatrue\fi}%
545   \cdp@list
546   \if@tempswa
547     \expandafter\SetSymbolFont@
548       \csname mv@#2\expandafter\endcsname\csname#3/#4/#5/#6\expandafter
549       \endcsname \csname sym#1\endcsname
550   \else
551     \@latex@error{Encoding scheme '#3' unknown}\@eha
552   \fi
553 }
554 \only\SetSymbolFont
555 </2ekernel | latexrelease>
556 <latexrelease>\EndIncludeInRelease
557 <latexrelease>\IncludeInRelease{0000/00/00}%
558 <latexrelease>           {\SetSymbolFont}{maybe drop m}%
559 <latexrelease>
560 <latexrelease>\let\SetSymbolFont\SetSymbolFont@m@dropped
561 <latexrelease>\let\SetSymbolFont@m@dropped\undefined
562 <latexrelease>
563 <latexrelease>\EndIncludeInRelease
564 <*2ekernel>

```

(End of definition for \SetSymbolFont.)

```

\SetSymbolFont@
565 \def\SetSymbolFont@#1#2#3{%
566   \expandafter\in@\expandafter#1\expandafter{\version@list}%
567   \ifin@%
568     \expandafter\in@\expandafter#3\expandafter{\group@list}%
569   \ifin@%
570     \begingroup
571       \expandafter\get@cdp\string#2\@nil\reserved@a
572       \toks@{}%
573       \def\install@mathalphabet##1##2{%
574         \addto@hook\toks@{\install@mathalphabet##1##2}%
575       }%
576       \def\getanddefine@fonts##1##2{%
577         \ifnum##1=##2%
578           \addto@hook\toks@{\getanddefine@fonts##2}%
579           \expandafter\get@cdp\string##2\@nil\reserved@b
580           \ifx\reserved@a\reserved@b\else
581             \font@info{Encoding '\reserved@b' has changed
582               to '\reserved@a' for symbol font\MessageBreak
583               '\expandafter@gobblefour\string#3' in the
584               math version '\expandafter
585               \@gobblefour\string#1'}%
586           \fi
587           \font@info{%
588             Overwriting symbol font
589             '\expandafter@gobblefour\string#3' in

```

```

590         version '\expandafter
591         \@gobblefour\string#1'\MessageBreak
592         \@spaces \expandafter\@gobble\string##2 -->
593             \expandafter\@gobble\string#2}%
594     \else
595         \addto@hook{\toks@{\getanddefine@fonts##1##2}%
596         \fi}%
597     #1%
598     \xdef#1{\the\toks@}%
599     \endgroup
600   \else
601     \@latex@error{Symbol font '\expandafter\@gobblefour\string#3'
602                   not defined}\@eha
603   \fi
604 \else
605   \@latex@error{Math version '\expandafter\@gobblefour\string#1'
606                 is not
607                 defined}{You probably misspelled the name of the math
608                 version.^^JOr you have to specify an additional package.}%
609 \fi
610 }
611 \onlypreamble\SetSymbolFont@
```

(End of definition for \SetSymbolFont@.)

```
\get@cdp
612 \def\get@cdp#1#2/#3@nil#4{\def#4{#2}}
613 \onlypreamble\get@cdp
```

(End of definition for \get@cdp.)

\DeclareMathAlphabet

```

614 \def\DeclareMathAlphabet#1#2#3#4#5{%
615   \tempswafalse
616   \edef\reserved@b{#2}%
617   \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
618     \ifx\reserved@b\reserved@c \tempswatrue\fi}%
619   \cdp@list
620   \if@tempswa
621     \expandafter\ifx
622     \csname\expandafter\@gobble\string#1\endcsname
623     \relax
624     \new@mathalphabet#1{#2}{#3}{#4}{#5}%
625   \else
```

Check if it is already a math alphabet.

```

626   \edef\reserved@a{\noexpand\in@\{\string\select@group\}%
627     {\expandafter\meaning\csname \expandafter
628       \@gobble\string#1\space\endcsname}\}%
629   \reserved@a
630   \ifin@
631     \font@info{Redeclaring math alphabet \string#1}%
632     \def\version@elt##1{%
633       \expandafter\SetMathAlphabet@\\expandafter
634         ##1\csname#2/#3/#4/#5\expandafter\endcsname}
```

```

635          \csname M@#2\expandafter\endcsname
636          \csname \expandafter\gobble\string#1\space\endcsname#1}%
637          \version@list
638      \else

```

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

```

639          \edef\reserved@a{\noexpand\in@\{\string\use@mathgroup\}%
640              \expandafter\meaning\csname \expandafter
641                  \gobble\string#1\space\endcsname\}%
642          \reserved@a
643          \ifin@

```

In that case overwriting is simple since there is nothing inserted in the math version macros.

```

644          \font@info{Redeclaring math alphabet \string#1}%
645          \new@mathalphabet#1{#2}{#3}{#4}{#5}%

```

Otherwise panic.

```

646          \else
647              \@latex@error{Command '\string#1' already defined}\@eha
648          \fi
649          \fi
650          \fi
651          \else
652              \@latex@error{Encoding scheme '#2' unknown}\@eha
653          \fi
654      }
655      \onlypreamble\DeclareMathAlphabet

```

(End of definition for `\DeclareMathAlphabet`.)

```

\new@mathalphabet
556 \def\new@mathalphabet#1#2#3#4#5{%
557     \toks@\expandafter{\alpha@list}%
558     \edef#1{\expandafter\noexpand\csname \expandafter
559             \gobble\string#1\space\endcsname
560             \if/#5/%
561                 \noexpand\no@alphabet@error
562                 \noexpand\no@alphabet@error
563             \else
564                 \expandafter\noexpand\csname M@#2\endcsname
565                 \expandafter\noexpand\csname#2/#3/#4/#5\endcsname
566             \fi
567         }%
568     \toks2\expandafter{#1}%
569     \edef\alpha@list{\the\toks@\noexpand\alpha@elt\the\toks2}%
570     \def\version@elt##1{\toks@\expandafter{##1}%
571         \edef##1{\the\toks@\install@mathalphabet
572             \expandafter\noexpand
573                 \csname \expandafter\gobble
574                     \string#1\space\endcsname
575                     \if/#5/%
576                         \noexpand\no@alphabet@error
577                         \noexpand#1%
578                     \else

```

```

679                               \noexpand\select@group\the\toks2
680                               \fi}\}%
681                         }%
682 \version@list
683 \expandafter\edef\csname \expandafter\@gobble
684           \string#1\space\endcsname{\if/#5/%
685             \noexpand\no@alphabet@error
686             \noexpand#1%
687           \else
688             \noexpand\select@group\the\toks2
689             \fi}\}%
690 \edef#1{\noexpand\protect
691   \expandafter\noexpand\csname \expandafter
692     \@gobble\string#1\space\endcsname}%
693 }
694 \onlypreamble\new@mathalphabet

```

(End of definition for `\new@mathalphabet`.)

`\SetMathAlphabet`

```

695 \def\SetMathAlphabet#1#2#3#4#5#6{%
696   \tempswafalse
697   \edef\reserved@b{#3}%
698   \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
699     \ifx\reserved@b\reserved@c \tempswatrue\fi}%
700 \cdp@list
701 \if@tempswa
702   \expandafter\SetMathAlphabet@
703     \csname mv@#2\expandafter\endcsname\csname#3/#4/#5/#6\expandafter
704     \endcsname \csname M@#3\expandafter\endcsname
705     \csname \expandafter\@gobble\string#1\space\endcsname#1%
706   \else
707     \latext@error{Encoding scheme '#3' unknown}\eha
708   \fi
709 }
710 \onlypreamble\SetMathAlphabet

```

(End of definition for `\SetMathAlphabet`.)

`\SetMathAlphabet@`

```

711 \def\SetMathAlphabet@#1#2#3#4#5{%
712   \expandafter\in@\expandafter#1\expandafter{\version@list}%
713   \ifin@
714     \expandafter\in@\expandafter#4\expandafter{\alpha@list}%
715   \ifin@
716     \begingroup
717       \toks@{}%
718       \def\getanddefine@fonts##1##2{%
719         \addto@hook\toks@{\getanddefine@fonts##1##2}%
720       }%
721       \def\reserved@c##1##2##3##4{%
722         \expandafter\@gobble\string##4}%
723       \def\install@mathalphabet##1##2{%
724         \ifx##1##2%
725           \addto@hook\toks@

```

```

726          {\install@mathalphabet#4{\select@group#4#3#2}}%
727  \@font@info{Overwriting math alphabet
728      'string#5' in version '\expandafter
729      \gobblefour\string#1'\MessageBreak
730      \spaces \reserved@c##2 -->
731          \expandafter\gobble\string#2}%
732  \else
733      \addto@hook\toks@{\install@mathalphabet##1{##2}}%
734  \fi
735  }%
736  #1%
737  \xdef#1{\the\toks@}%
738  \endgroup
739 \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.

```

740      \edef\reserved@a{%
741          \noexpand\in@\{`string`\use@mathgroup}{\meaning#4}}%
742  \reserved@a
743  \ifin@
744      \def\reserved@b##1\use@mathgroup##2##3{%
745          \def\reserved@b{##3}\def\reserved@c{##2}}%
746      \expandafter\reserved@b#4%
747  \begingroup
748      \def\install@mathalphabet##1##2{%
749          \addto@hook\toks@{\install@mathalphabet##1{##2}}%
750      }%
751      \def\getanddefine@fonts##1##2{%
752          \addto@hook\toks@{\getanddefine@fonts##1##2}}%
753      \ifnum##1=\reserved@b
754          \expandafter
755          \addto@hook\expandafter\toks@
756          \expandafter{\expandafter\install@mathalphabet
757          \expandafter#4\expandafter
758              {\expandafter\select@group\expandafter
759                  #4\reserved@c##2}}%
760      \fi
761  }%
762  \def\version@elt##1{%
763      \toks@{}%
764      ##1%
765      \xdef##1{\the\toks@}%
766  }%
767  \version@list
768  \endgroup

```

Put it into the `\alpha@list` with default ‘error’

```

769      \expandafter\gdef\expandafter\alpha@list\expandafter
770          {\alpha@list
771              \alpha@elt #4\no@alphabet@error \no@alphabet@error}%
772              \gdef#4{\no@alphabet@error #5}%

```

Then call the internal setting routine again:

```

773     \SetMathAlphabet@{\#1}{\#2}{\#3}#4#5%
774     \else
775         \@latex@error{Command '\string#5' not defined as a
776                         math alphabet}%
777             {Use \noexpand\DeclareMathAlphabet to define it.}%
778         \fi
779     \fi
780 \else
781     \@latex@error{Math version '\expandafter\gobblefour\string#1'
782                     is not
783                     defined}{You probably misspelled the name of the math
784                     version.^JOr you have to specify an additional package.}%
785 \fi
786 }
787 \onlypreamble\SetMathAlphabet@
```

(End of definition for \SetMathAlphabet@.)

\DeclareMathAccent Could do with more checks like allowing single number in #4 lowercase in #4 etc

```

788 </2ekernel>
789 <*2ekernel | latexrelease>
790 <latexrelease>\IncludeInRelease{2019/10/01}%
791 <latexrelease>           {DeclareMathAccent}{Make math accents robust}%
792 \def\DeclareMathAccent#1#2#3#4{%
793     \expandafter\in@\csname sym#3\expandafter\endcsname
794         \expandafter{\group@list}%
795 \ifin@
796     \begingroup
797         \count\z@=#4\relax
798         \count\tw@\count\z@
799         \divide\count\z@\sixt@@n
800         \count@\count\z@
801         \multiply\count@\sixt@@n
802         \advance\count\tw@-\count@
803         \if\relax\noexpand#1% is command?
804             \edef\reserved@a{\noexpand\in@
805                 {\expandafter\gobble\string\mathaccent}
806                 {\expandafter\meaning
807                     \csname\expandafter\gobble\string#1\space\endcsname}%
808             \reserved@a
809             \ifin@
810                 \expandafter\let
811                     \csname\expandafter\gobble\string#1\space\endcsname
812                     \undefined
813                 \expandafter\set@mathaccent
814                     \csname sym#3\endcsname#1#2%
815                     {\hexnumber@\{\count\z@\}\hexnumber@\{\count\tw@\}}%
816                     \font@info{Redeclaring math accent \string#1}%
817             \else
818                 \expandafter\ifx
819                     \csname\expandafter\gobble\string#1\endcsname
820                     \relax
821                     \expandafter\set@mathaccent
822                     \csname sym#3\endcsname#1#2%
```

```

823          {\hexnumber@{\count\z@\hexnumber@{\count\tw@}}\%
824          \else
825              \@latex@error{Command ‘\string#1’ already defined}\@eha
826          \fi
827          \fi
828          \else
829              \@latex@error{Not a command name: ‘\noexpand#1’}\@eha
830          \fi
831      \endgroup
832      \else
833          \@latex@error{Symbol font ‘#3’ is not defined}\@eha
834      \fi
835  }
836 (/2ekernel | latexrelease)
837 <latexrelease>\EndIncludeInRelease
838 <latexrelease>\IncludeInRelease{0000/00/00}%
839 <latexrelease>                                {DeclareMathAccent}{Make math accents robust}%
840 <latexrelease>\def\DeclareMathAccent#1#2#3#4{%
841 <latexrelease>    \expandafter\in@{\csname sym#3\expandafter\endcsname
842 <latexrelease>        \expandafter{\group@list}%
843 <latexrelease>    \ifin@
844 <latexrelease>        \begingroup
845 <latexrelease>            \count\z@=\#4\relax
846 <latexrelease>            \count\tw@\count\z@
847 <latexrelease>            \divide\count\z@\sixt@on
848 <latexrelease>            \count@\count\z@
849 <latexrelease>            \multiply\count@\sixt@on
850 <latexrelease>            \advance\count\tw@-\count@
851 <latexrelease>            \if\relax\noexpand#1% is command?
852 <latexrelease>                \edef\reserved@a{\noexpand\in@
853 <latexrelease>                    {\expandafter\gobble\string\mathaccent}{\meaning#1}}%
854 <latexrelease>                \reserved@a
855 <latexrelease>                \ifin@
856 <latexrelease>                    \expandafter\set@mathaccent
857 <latexrelease>                        \csname sym#3\endcsname#1#2%
858 <latexrelease>                        {\hexnumber@{\count\z@\hexnumber@{\count\tw@}}\%
859 <latexrelease>                        \font@info{Redeclaring math accent \string#1}}%
860 <latexrelease>                \else
861 <latexrelease>                    \expandafter\ifx
862 <latexrelease>                        \csname\expandafter\gobble\string#1\endcsname
863 <latexrelease>                        \relax
864 <latexrelease>                            \expandafter\set@mathaccent
865 <latexrelease>                                \csname sym#3\endcsname#1#2%
866 <latexrelease>                                {\hexnumber@{\count\z@\hexnumber@{\count\tw@}}\%
867 <latexrelease>                            \else
868 <latexrelease>                                \@latex@error{Command ‘\string#1’ already defined}\@eha
869 <latexrelease>                            \fi
870 <latexrelease>                        \fi
871 <latexrelease>                    \else
872 <latexrelease>                        \@latex@error{Not a command name: ‘\noexpand#1’}\@eha
873 <latexrelease>                    \fi
874 <latexrelease>                \endgroup
875 <latexrelease>            \else
876 <latexrelease>                \@latex@error{Symbol font ‘#3’ is not defined}\@eha

```

```

877 〈\latexrelease〉 \fi
878 〈\latexrelease〉}
879 〈\latexrelease〉\EndIncludeInRelease
880 〈*2ekernel〉
881 \onlypreamble\DeclareMathAccent
(End of definition for \DeclareMathAccent.)
```

\set@mathaccent

```

882 〈/2ekernel〉
883 〈*2ekernel | \latexrelease〉
884 〈\latexrelease〉\IncludeInRelease{2019/10/01}%
885 〈\latexrelease〉 \set@mathaccent\makemath accents robust}%
886 \def\set@mathaccent#1#2#3#4{%
887   \xdef#2{\mathaccent"\mathchar@type#3\hexnumber@#1#4\relax}%
888   \MakeRobust#2%
889 }
890 \onlypreamble\set@mathaccent
891 〈/2ekernel | \latexrelease〉
892 〈\latexrelease〉\EndIncludeInRelease
893 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
894 〈\latexrelease〉 \set@mathaccent\makemath accents robust}%
895 〈\latexrelease〉
896 〈\latexrelease〉\def\set@mathaccent#1#2#3#4{%
897 〈\latexrelease〉 \xdef#2{\mathaccent"\mathchar@type#3\hexnumber@#1#4\relax}%
898 〈\latexrelease〉
899 〈\latexrelease〉\EndIncludeInRelease
900 〈*2ekernel〉
```

(End of definition for \set@mathaccent.)

\DeclareMathSymbol

```

901 \def\DeclareMathSymbol#1#2#3#4{%
902   \expandafter\in@\csname sym#3\expandafter\endcsname
903   \expandafter{\group@list}%
904 \ifin@
905   \begingroup
906     \count\z@=#4\relax
907     \count\tw@\count\z@
908     \divide\count\z@\sixt@@n
909     \count@\count\z@
910     \multiply\count@\sixt@@n
911     \advance\count\tw@-\count@
912     \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.

```

913   \edef\reserved@b{\expandafter\noexpand
914   \csname\expandafter\@gobble\string#1\space\endcsname}%
```

Test both #1 and #1_l for containing mathchar.

```

915   \edef\reserved@a
916   {\noexpand\in@\{\expandafter\@gobble\string\mathchar\}%
917   {\meaning#1\expandafter\meaning\reserved@b}\}%
918 \reserved@a
```

Drop #1_↓ in case it was defined before.

```
919     \global\expandafter\let\reserved@b\@undefined
920     \ifin@
921         \expandafter\set@mathsymbol
922             \csname sym#3\endcsname#1#2%
923             {\hexnumber@\{\count\z@\}\hexnumber@\{\count\tw@\}}%
924             \font@info{Redeclaring math symbol \string#1}%
925     \else
926         \expandafter\ifx
927             \csname\expandafter\gobble\string#1\endcsname
928             \relax
929             \expandafter\set@mathsymbol
930                 \csname sym#3\endcsname#1#2%
931                 {\hexnumber@\{\count\z@\}\hexnumber@\{\count\tw@\}}%
932             \else
933                 \@latex@error{Command ‘\string#1’ already defined}\@eha
934             \fi
935         \fi
936     \else
937         \expandafter\set@mathchar
938             \csname sym#3\endcsname#1#2
939             {\hexnumber@\{\count\z@\}\hexnumber@\{\count\tw@\}}%
940         \fi
941     \endgroup
942 \else
943     \@latex@error{Symbol font ‘#3’ is not defined}\@eha
944 \fi
945 }
946 \onlypreamble\DeclareMathSymbol
```

(End of definition for `\DeclareMathSymbol`.)

```
\set@mathchar
947 \def\set@mathchar#1#2#3#4{%
948     \global\mathcode‘#2=”\mathchar@type#3\hexnumber@#1#4\relax}
949 \onlypreamble\set@mathchar
```

(End of definition for `\set@mathchar`.)

```
\set@mathsymbol
950 \def\set@mathsymbol#1#2#3#4{%
951     \global\mathchardef#2”\mathchar@type#3\hexnumber@#1#4\relax}
952 \onlypreamble\set@mathsymbol
```

(End of definition for `\set@mathsymbol`.)

```
953 \% \def\mathsymbol#1#2#3#4{%
954 %   \tempcnta=#3\relax
955 %   \tempcntb\tempcnta
956 %   \divide\tempcnta\sixt@n
957 %   \count@\tempcnta
958 %   \multiply\count@\sixt@n
959 %   \advance\tempcntb-\count@
960 %   \mathchar”\mathchar@type#1\hexnumber@#2%
961 %           \hexnumber@\tempcnta\hexnumber@\tempcntb\relax}
```

```

962 %
963 \%def\DeclareMathAlphabetCharacter#1#2#3{%
964 % \DeclareMathSymbol{#1}7{#2}{#3}{}

\DeclareMathDelimiter

965 \def\DeclareMathDelimiter#1{%
966   \if\relax\noexpand#1%
967     \expandafter\@DeclareMathDelimiter
968   \else
969     \expandafter\@xxDeclareMathDelimiter
970   \fi
971   #1}
972 \onlypreamble\DeclareMathDelimiter

(End of definition for \DeclareMathDelimiter.)

```

(End of definition for \DeclareMathDelimiter.)

`\@xDeclarMathDelimiter` 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.

973 \def\@xDeclarMathDelimiter#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!

```
974 \begingroup  
975   \let\mathalpha\mathord  
976   \ifnum7=\mathchar@type{#2}%  
977     \endgroup
```

If this branch is taken we have old syntax (5 arguments).

```
978     \expandafter\@firstofone  
979     \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.

```
980     \endgroup  
981     \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.

```
982     \expandafter\@firstoftwo
983     \fi
984     {\@xDeclareMathDelimiter#1{#2}{#3}{#4}}
985 \onlypreamble\@xDeclareMathDelimiter
```

(End of definition for \c@xxDeclarMathDelimiter.)

\@DeclareMathDelimiter

```
986 \def\@DeclareMathDelimiter#1#2#3#4#5#6{%
987   \expandafter\in@\csname sym#3\expandafter\endcsname
988   \expandafter{\group@list}%
989 \ifin@
```

```

990 \expandafter\in@{\csname sym#5\expandafter\endcsname
991   \expandafter{\group@list}%
992 \ifin@
993   \begingroup
994     \count\z@=#4\relax
995     \count\tw@\count\z@
996     \divide\count\z@\sixt@@n
997     \count@\count\z@
998     \multiply\count@\sixt@@n
999     \advance\count\tw@-\count@
1000     \edef\reserved@c{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
1001 %
1002   \count\z@=#6\relax
1003   \count\tw@\count\z@
1004   \divide\count\z@\sixt@@n
1005   \count@\count\z@
1006   \multiply\count@\sixt@@n
1007   \advance\count\tw@-\count@
1008   \edef\reserved@d{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
1009 %
1010   \edef\reserved@a{\noexpand\in@
1011     {\expandafter\@gobble\string\delimiter}{\meaning#1}}%
1012 \reserved@a
1013 \ifin@
1014   \expandafter\set@mathdelimiter
1015     \csname sym#3\expandafter\endcsname
1016     \csname sym#5\endcsname#1#2%
1017     \reserved@c\reserved@d
1018     \@font@info{Redeclaring math delimiter \string#1}%
1019 \else
1020   \expandafter\ifx
1021     \csname\expandafter\@gobble\string#1\endcsname
1022     \relax
1023     \expandafter\set@mathdelimiter
1024       \csname sym#3\expandafter\endcsname
1025       \csname sym#5\endcsname#1#2%
1026       \reserved@c\reserved@d
1027     \else
1028       \@latex@error{Command '\string#1' already defined}\@eha
1029     \fi
1030   \fi
1031   \endgroup
1032 \else
1033   \@latex@error{Symbol font '#5' is not defined}\@eha
1034   \fi
1035 \else
1036   \@latex@error{Symbol font '#3' is not defined}\@eha
1037   \fi
1038 }
1039 \onlypreamble\@DeclareMathDelimiter

```

(End of definition for \@DeclareMathDelimiter.)

\@xDeclareMathDelimiter

```

1040 \def\@xDeclareMathDelimiter#1#2#3#4#5{%
1041   \expandafter\in@\csname sym#2\expandafter\endcsname
1042   \expandafter{\group@list}%
1043 \ifin@
1044   \expandafter\in@\csname sym#4\expandafter\endcsname
1045   \expandafter{\group@list}%
1046 \ifin@
1047   \begingroup
1048     \count\z@=#3\relax
1049     \count\tw@\count\z@
1050     \divide\count\z@\sixt@@n
1051     \count@\count\z@
1052     \multiply\count@\sixt@@n
1053     \advance\count\tw@-\count@
1054     \edef\reserved@c{\hexnumber@\{\count\z@\}\hexnumber@\{\count\tw@\}}%
1055   %
1056     \count\z@=#5\relax
1057     \count\tw@\count\z@
1058     \divide\count\z@\sixt@@n
1059     \count@\count\z@
1060     \multiply\count@\sixt@@n
1061     \advance\count\tw@-\count@
1062     \edef\reserved@d{\hexnumber@\{\count\z@\}\hexnumber@\{\count\tw@\}}%
1063   \expandafter\set@mathdelimiter
1064     \csname sym#2\expandafter\endcsname\csname sym#4\endcsname#1%
1065     \reserved@c\reserved@d
1066   \endgroup
1067 \else
1068   \@latex@error{Symbol font ‘#4’ is not defined}\@eha
1069 \fi
1070 \else
1071   \@latex@error{Symbol font ‘#2’ is not defined}\@eha
1072 \fi
1073 }
1074 \onlypreamble\@xDeclareMathDelimiter

```

(End of 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)

```

1075 </2ekernel>
1076 <*2ekernel | latexrelease>
1077 <latexrelease>\IncludeInRelease{2019/10/01}%
1078 <latexrelease>          {\set@mathdelimiter}{make delimiters robust}%
1079 \def\set@mathdelimiter#1#2#3#4#5#6{%

```

We use `\protected` not `\MakeRobust` so that `\bigl\lfloor` etc. works inside the argument of `\protected@edef`.

```

1080 \protected
1081 \xdef#3{\delimiter"\mathchar@type#4\hexnumber@#1#5%
1082   \hexnumber@#2#6 }%
1083 % \MakeRobust#3%
1084 }

```

```

1085  \Oonlypreamble\set@mathdelimiter
1086  </2ekernel | latexrelease>
1087  <latexrelease>\EndIncludeInRelease
1088  <latexrelease>\IncludeInRelease{0000/00/00}%
1089  <latexrelease>          {\set@mathdelimiter}{make delimiters robust}%
1090  <latexrelease>
1091  <latexrelease>\def\set@mathdelimiter#1#2#3#4#5#6{%
1092  <latexrelease>  \xdef#3{\delimiter"\mathchar@type#4\hexnumber@#1#5%
1093  <latexrelease>    \hexnumber@#2#6 }
1094  <latexrelease>
1095  <latexrelease>\EndIncludeInRelease
1096  {*2ekernel}

```

(End of definition for `\set@mathdelimiter`.)

```

\set@@mathdelimiter
1097  \def\set@@mathdelimiter#1#2#3#4#5{%
1098  <global\delcode'3="\hexnumber@#1#4\hexnumber@#2#5\relax}
1099  \Oonlypreamble\set@@mathdelimiter

```

(End of definition for `\set@@mathdelimiter`.)

`\DeclareMathRadical`

```

1100  \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!

```

1101  \expandafter\ifx
1102  <csname\expandafter\@gobble\string#1\endcsname
1103  <\relax
1104  <\let#1\radical
1105  <\fi
1106  \edef\reserved@a{\noexpand\in@
1107  <\expandafter\@gobble\string\radical}{\meaning#1}%
1108  \reserved@a
1109  \ifin@
1110  <\expandafter\in@\csname sym#2\expandafter\endcsname
1111  <\expandafter{\group@list}%
1112  \ifin@
1113  <\expandafter\in@\csname sym#4\expandafter\endcsname
1114  <\expandafter{\group@list}%
1115  \ifin@
1116  <\begingroup
1117  <\count\z@=#3\relax
1118  <\count\tw@\count\z@
1119  <\divide\count\z@\sixt@n
1120  <\count@\count\z@
1121  <\multiply\count@\sixt@n
1122  <\advance\count\tw@-\count@
1123  <\edef\reserved@c{%
1124  <\hexnumber@{\count\z@}\hexnumber@{\count\tw@}%
1125  <\count\z@=#5\relax
1126  <\count\tw@\count\z@
1127  <\divide\count\z@\sixt@n
1128  <\count@\count\z@

```

```

1129          \multiply\count@\sixt@n
1130          \advance\count@\tw@-\count@
1131          \edef\reserved@d{%
1132              \hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
Coded inline instead of using \set@mathradical
1133  %
1134  %           \expandafter\set@mathradical
1135  %           \csname sym#2\expandafter\endcsname
1136  %           \csname sym#4\endcsname#1%
1137  %           \reserved@c\reserved@d
1138      \xdef#1{\radical"\expandafter\hexnumber@
1139          \csname sym#2\endcsname\reserved@c
1140          \expandafter\hexnumber@
1141          \csname sym#4\endcsname\reserved@d
1142          \relax}%
1143      \endgroup
1144  \else
1145      \@latex@error{Symbol font '#4' is not defined}\@eha
1146  \fi
1147  \else
1148      \@latex@error{Symbol font '#2' is not defined}\@eha
1149  \fi
1150  \else
1151      \@latex@error{Command '\string#1' already defined}\@eha
1152  \fi
1153 \onlypreamble\DeclareMathRadical

```

(End of 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
1154 \let\mathalpha\relax

(End of definition for \mathalpha.)

\mathchar@type

```

1155 \def\mathchar@type#1{%
1156   \ifodd 2#11 #1\else          % is this non-negative number?
1157     \ifx#1\mathord 0\else
1158       \ifx#1\mathop 1\else
1159         \ifx#1\mathbin 2\else
1160           \ifx#1\mathrel 3\else
1161             \ifx#1\mathopen 4\else
1162               \ifx#1\mathclose 5\else
1163                 \ifx#1\mathpunct 6\else
1164                   7%          % anything else is variable ord
1165                   \fi
1166                 \fi
1167               \fi
1168             \fi
1169           \fi

```

```

1170      \fi
1171      \fi
1172  \fi}
1173 \onlypreamble\mathchar@type

```

(End of definition for \mathchar@type.)

\DeclareSymbolFontAlphabet

```

1174 \def\DeclareSymbolFontAlphabet#1#2{%
1175   \expandafter\DeclareSymbolFontAlphabet@
1176   \csname \expandafter\gobble\string#1\space\endcsname{#2}#1}
1177 \onlypreamble\DeclareSymbolFontAlphabet

```

(End of definition for \DeclareSymbolFontAlphabet.)

\DeclareSymbolFontAlphabet@

```

1178 \def\DeclareSymbolFontAlphabet@#1#2#3{%

```

We use the switch \if@tempswa to decide if we can declare this symbol font alphabet.

```

1179 \if@tempswa true

```

First check if #2 is known to be a symbol font

```

1180 \expandafter\in@\csname sym#2\expandafter\endcsname
1181   \expandafter{\group@list}%
1182 \ifin@
```

Check if #1 is defined as a math alphabet defined via \DeclareMathAlphabet:

```

1183 \expandafter\in@\expandafter#1\expandafter{\alpha@list}%
1184 \ifin@
```

If so remove it from the \alpha@list and from all math version macros.

```

1185   @font@info{Redeclaring math alphabet \string#3}%
1186   \toks@{}%
1187   \def\alpha@elt##1##2##3{%
1188     \ifx##1#1\else\addto@hook\toks@{\alpha@elt##1##2##3}\fi}%
1189   \alpha@list
1190   \xdef\alpha@list{\the\toks@}%

```

Now we loop over all versions and remove the math alphabet:

```

1191 \def\version@elt##1{%
1192   \begingroup
1193   \toks@{}%
1194   \def\getanddefine@fonts####1####2{%
1195     \addto@hook\toks@{\getanddefine@fonts####1####2}}%
1196   \def\install@mathalphabet####1####2{%
1197     \ifx####1#1\else
1198       \addto@hook\toks@{\install@mathalphabet
1199         ####1{####2}}\fi}%
1200     ##1%
1201     \xdef##1{\the\toks@}%
1202   \endgroup
1203 }%
1204 \version@list
1205 \else

```

If #3 is not defined as a math alphabet check if it is defined at all:

```
1206     \expandafter\ifx
1207     \csname\expandafter\gobble\string#1\space\endcsname
1208     \relax
```

If it is undefined, fine otherwise check if it is a math alphabet defined via `\DeclareSymbolFontAlphabet`:

```
1209     \else
1210     \edef\reserved@a{%
1211         \noexpand\in@\{\string\use@mathgroup\{\meaning#1}\}%
1212         \reserved@a
1213         \ifin@
1214             \font@info{Redeclaring math alphabet \string#3}%
1215         \else
```

Since the command #3 is defined to be something which is not a math alphabet we have to skip redefining it.

```
1216         \tempswafalse
1217         \latex@error{Command '\string#3' already defined}\@eha
1218     \fi
1219     \fi
1220     \fi
1221 \else
```

Since the symbol font is not known we better skip defining this alphabet.

```
1222     \tempswafalse
1223     \latex@error{Unknown symbol font '#2'}\@eha
1224 \fi
1225 \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.

```
1226 \def\group@elt##1##2{%
1227     \expandafter\ifx\csname sym#2\endcsname##1%
1228     \expandafter\reserved@a\string##2\@nil
1229     \fi}%
1230 \def\reserved@a##1##2##3\@nil{%
1231     \def\reserved@a{##2}%
1232 \group@list
1233 \toks@{\relax\ifmmode \else \non@alpherr#1\fi}%
1234 \edef#1{\the\toks@
1235     \noexpand\use@mathgroup
1236     \expandafter\noexpand\csname M@\reserved@a\endcsname
1237     \csname sym#2\endcsname}%
1238 \def#3{\protect#1}%
1239 \fi
1240 }
1241 \onlypreamble\DeclareSymbolFontAlphabet@
1242 {/2ekernel}
```

(End of definition for \DeclareSymbolFontAlphabet@.)

File A

ltfssini.dtx

This file contains the top level L^AT_EX interface to the font selection scheme commands. See other parts of the L^AT_EX distribution, or *The L^AT_EX Companion* for higher level documentation of these commands.

1 NFSS Initialization

Finally, there are six commands that are to be used in L^AT_EX 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^AT_EX 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^AT_EX’s standard setting for `\bfdefault` didn’t work with such fonts, but if it got changed to produce `b`, then that didn’t work with Computer Modern if the fonts got combined (e.g., using Computer Modern Typewriter with such fonts).

The solution back then was to provide substitution rules in the font .fd such that if a `bx` series got requested the `b` series got used. While this works in that particular case, it isn’t a very general solution. For example, if you happen to have a font family that has several weights you may want to typeset the whole document in a somewhat lighter or darker font but if you then modify `\mddefault` to allow for this, then of course your change only works with that particular family but not with the typewriter or sans serif family you also want to use.

A better solution was provided by the `mweights` package by Bob Tennent that offers defaults on the level of the three main font families in the document: for “rm”, “sf” and “tt” so that font packages could define defaults for the sans serif document font by providing `\bfseries@sf` which then was used when `\bfseries` got executed and the current family was the `\sffamily`.

`\DeclareFontSeriesDefault`

We now support this concept directly from within L^AT_EX 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!³²

`\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

```

Adding `\@empty` allows us to detect if the default gets redefined with `\renewcommand` or `\def` by the user.

```

44      \expandafter\def
45      \csname #2default\endcsname{\#3\@empty}%
46      \expandafter\def
47      \csname #2default@previous\endcsname{\#3\@empty}%
48  \else
49      \@latex@error{Wrong syntax for \string\DeclareFontSeriesDefault}{%
50      Mandatory first argument must be 'md' or 'bf'.}
51  \fi

```

Optional argument given, set up specific default.

```

52  \else
53      \ifcsname #2series@\#1\endcsname            % supported are
54      \ifcsname #3series@\#1\endcsname
55      \expandafter\edef
56      \csname #2series@\#1\endcsname{\#3}%

```

If the interface is used we remove the frozen kernel default. This way, we know that something was explicitly set up (even if the setup has the same value as the default).

```

57      \expandafter\let
58      \csname #2series@\#1@kernel\endcsname\@undefined
59  \else
60      \@latex@error{Wrong syntax for \string\DeclareFontSeriesDefault}{%
61      Optional argument must be 'rm', 'sf', or 'tt'. \MessageBreak
62      Mandatory first argument must be 'md' or 'bf'.}
63  \fi
64 \fi
65 }

```

³²I 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 of 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{}}{2020/02/02}
126  \begin{macro}{\textrm}{\textrm{}}{mdseries@rm}{Custom series}
```

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

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

`\bfseries@sf` To account for the fact that by default we typeset in CM or LM we set up the `\bfseries@..` defaults to use `bx` instead.

This means that it behaves like before because if the default fonts are used then `\bfseries@rm` etc kick in and make `\textbf` use `bx`. However, if the font gets changed then `\bfdefault` will get used.

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

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

```
131 \let\bfsf@rm@kernel\bfsf@rm
132 \let\bfsf@sf@kernel\bfsf@sf
133 \let\bfsf@tt@kernel\bfsf@tt
```

The default for the medium series is `m` and this will be interpreted as resetting both weight and width. To reset only one of them the virtual value `?m` and `m?` are available.

```
134 \def\mdseries@rm{m}
135 \def\mdseries@sf{m}
136 \def\mdseries@tt{m}
```

(*End of 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 \begin{macro}{\series@change@debug}\typeout
139 \begin{macro}{\series@change@debug}\@gobble
140 \begin{macro}{\series@change@debug}
```

(*End of 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 of 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 of 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 of 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 of 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 of 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 of 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 of 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.

`\bf@def@ult`

```

397 \def\expand@font@defaults{%
398   \edef\rmdef@ult{\rmdefault}%
399   \edef\sfdef@ult{\sfdefault}%
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 of 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 of definition for `\rmfamily`, `\sffamily`, and `\ttfamily`.)

```
rmfamily    Declare the hooks used above.  
sffamily    420 \NewHook{rmfamily}  
ttfamily    421 \NewHook{sffamily}  
normalfont  422 \NewHook{ttfamily}  
expand@font@defaults 423 \NewHook{normalfont}  
bfseries    424 \NewHook{expand@font@defaults}  
bfseries/defaults 425 \NewHook{bfseries}  
mdseries    426 \NewHook{bfseries/defaults}  
mdseries/defaults 427 \NewHook{mdseries}  
428 \NewHook{mdseries/defaults}
```

(End of definition for `rmfamily` and others.)

`\@rmfamilyhook` These four hooks have legacy versions used in 2020/02/02 so we should support them
`\@sffamilyhook` until they aren't any longer used.

By default the hooks do nothing.

```
\@defaultfamilyhook 429 \let\@rmfamilyhook\@empty  
                      430 \let\@sffamilyhook\@empty  
                      431 \let\@ttfamilyhook\@empty  
                      432 \let\@defaultfamilyhook\@empty %FMi sort out
```

(End of definition for \@rmfamilyhook and others.)

```
433 </2ekernel | latexrelease>
434 <|latexrelease>\EndIncludeInRelease
435 <|latexrelease>\IncludeInRelease{2020/02/02}%
436 <|latexrelease>           {\expandafter\font@defaults}{Custom series with hooks}%
437 <|latexrelease>
438 <|latexrelease>\def\expandafter{\font@defaults{%
439 <|latexrelease> \edef\rmdefault{\font@ult{\rmdefault}}%
440 <|latexrelease> \edef\sffont{\font@ult{\sffont}}%
441 <|latexrelease> \edef\ttfont{\font@ult{\ttfont}}%
442 <|latexrelease> \edef\bfseries{\font@ult{\bfseries}}%
443 <|latexrelease> \edef\mdseries{\font@ult{\mdseries}}%
444 <|latexrelease> \edef\fam{\font@ult{\fam}}%
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 of 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 of 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 of 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^AT_EX 2 _{ε} 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^AT_EX 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.³³ 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^AT_EX.

```
548 \let\emfontdeclare@clist\empty
```

(End of 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 of 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 }
```

³³ 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 of 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 of 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 of definition for \emforce and \em@force.)

\em \eminnershape These are the older definitions for \em, prior to 2020.

We also have to define the *emphasize* font change command (i.e. \em). This command will look is the current font is sloped (i.e. has a positive \fontdimen1) and will then select either \upshape or \itshape.

```
577 <latexrelease>\IncludeInRelease[2015/01/01]{\DeclareEmphSequence}{Nested emph}%
```

```
578 <latexrelease>\let\DeclareEmphSequence\@undefined
```

```
579 <latexrelease>\let\emfontdeclare@clist\@undefined
```

```
580 <latexrelease>\let\emreset\@undefined
```

```
581 <latexrelease>\let\do@emfont@update\@undefined
```

```

582 〈\latexrelease〉\let\emforce\@undefined
583 〈\latexrelease〉\let\em@force\@undefined
584 〈\latexrelease〉
585 〈\latexrelease〉\DeclareRobustCommand\em
586 〈\latexrelease〉      {＼@nomath\em \ifdim \fontdimen\@ne\font >\z@
587 〈\latexrelease〉                           \eminnershape \else \itshape \fi}%
588 〈\latexrelease〉\EndIncludeInRelease
589 〈\latexrelease〉
590 〈\latexrelease〉\IncludeInRelease{0000/00/00}{\DeclareEmphSequence}{Nested emph}%
591 〈\latexrelease〉\DeclareRobustCommand\em
592 〈\latexrelease〉      {＼@nomath\em \ifdim \fontdimen\@ne\font >\z@
593 〈\latexrelease〉                           \upshape \else \itshape \fi}%
594 〈\latexrelease〉\let\eminnershape\@undefined
595 〈\latexrelease〉\EndIncludeInRelease
596 〈*2ekernel〉

```

(End of definition for `\em` and `\eminnershape`.)

`\not@math@alphabet` This function generates an error message when it is called in math mode. The same function should be defined in `newfont.sty`.

```

597 \def\not@math@alphabet#1#2{%
598   \relax
599   \ifmmode
600     \@latex@error{Command \noexpand#1 invalid in math mode}%
601     {%
602       Please
603       \ifx#2\relax
604         define a new math alphabet^{#1}%
605         if you want to use a special font in math mode%
606       \else

```

We have to a `\noexpand` below to prevent expansion of #2. In case of #1 we can omit this (due to the current definition of robust commands since they do come out right there :-).

```

607     use the math alphabet \noexpand#2 instead of
608     the #1 command%
609   \fi
610   .
611 }%
612 \fi}

```

(End of definition for `\not@math@alphabet`.)

Finally we provide two abbreviations to switch to the L^AT_EX 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^AT_EX'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 of 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 of definition for \symbol.)
```

3.2 Miscellaneous

\@setfontsize This abbreviation is used by L^AT_EX'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 of definition for \@setfontsize and \@setsizesize.)
```

\hexnumber@ To set up L^AT_EX'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 of 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 of 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 of definition for \copyright.)

\normalfont \reset@font The macro \reset@font is used in L^AT_EX 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^AT_EX 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 of 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^AT_EX fonts which are not automatically included in the base version of the font selection since these fonts contain only a few characters which are also included in the AMS fonts so anybody who is using these fonts doesn't need them. But for compatibility reasons we will define these symbols.

```

720 \def\not@base#1{\@latex@error
721   {Command \noexpand#1 not provided in base LATEX2e}%
722   {Load the latexsym or the amsfonts package to
723    define this symbol}}
724 \def\mho{\not@base\mho}
725 \def\Join{\not@base\Join}
726 \def\Box{\not@base\Box}
727 \def\Diamond{\not@base\Diamond}
728 \def\leadsto{\not@base\leadsto}
729 \def\sqsubset{\not@base\sqsubset}
730 \def\sqsupset{\not@base\sqsupset}
731 \def\lhd{\not@base\lhd}
732 \def\unlhd{\not@base\unlhd}
733 \def\rhd{\not@base\rhd}
734 \def\unrhd{\not@base\unrhd}

```

We now initialize all variables set by `\DeclareErrorFont`. These values are not really important since they will be overwritten later on by the definition in `fontdef.ltx`.

However, if `fontdef.cfg` is corrupted then at least a hopefully suitable error font is present.

```

735 \DeclareErrorFont{OT1}{cmr}{m}{n}{10} %% don't modify this setting
736                                     %% overwrite it in fontdef.cfg
737                                     %% if necessary

```

We also set some default values for `\f@family` etc. Note that we don't yet have any encodings that comes later. In the past this was implicitly done by `\DeclareErrorFont`.

```

738 \fontfamily{cmr}

```

Previously the default values for series and shape were set by calling `\fontseries` and `\fontshape`, but their action is now delayed until `\selectfont` which isn't called inside the format (to avoid unnecessarily loading a font that may never get used). We therefore have to set `\f@series` and `\f@shape` directly instead.

```

739 \def\f@series{m}          % \fontseries{m}
740 \def\f@shape{n}           % \fontshape{n}
741 \fontsize{10}{10}

```

The initial `fontenc` package load list. This will get overwritten in `fonttext` and is only provided in case an old `fonttext.cfg` does not define the command:

```

742 \def@\fontenc@load@list{\@elt{T1,OT1}}

```

We now load the customizable parts of NFSS.

```
743 \InputIfFileExists{fonttext.cfg}
744   {\typeout{=====
745     ^^^J%
746     Local config file fonttext.cfg used^ ^^J%
747     ^^^J%
748     =====}%
749   \def\@addtolist##1{\xdef\@filelist{\@filelist,##1}}%
750   }
751   {\input{fonttext.ltx}}
752 \let\@addtolist\@gobble
Ditto for math although I don't think that we will get a lot of customisation :-)
753 \InputIfFileExists{fontmath.cfg}
754   {\typeout{=====
755     ^^^J%
756     Local config file fontmath.cfg used^ ^^J%
757     ^^^J%
758     =====}%
759   \def\@addtolist##1{\xdef\@filelist{\@filelist,##1}}%
760   }
761   {\input{fontmath.ltx}}
762 \let\@addtolist\@gobble
```

Then we preload several fonts. This file might be customized *without* changing the behavior of the format (i.e. necessary font definitions will be loaded at runtime if they are not preloaded). This is done in the file `preload.ltx`.

```
763 \InputIfFileExists{preload.cfg}
764   {\typeout{=====
765     ^^^J%
766     Local config file preload.cfg used^ ^^J%
767     ^^^J%
768     =====}%
769   \def\@addtolist##1{\xdef\@filelist{\@filelist,##1}}%
770   }
771   {\input{preload.ltx}}
772 \let\@addtolist\@gobble
```

\seriesdefault After `\seriesdefault` got defined inside `fonttext.ltx` or a `.cfg` file overwriting it, we alter its value by appending `\empty` to it. This will vanish if expanded but allows us to check if the default gets altered (even to the same value) in the document preamble. All we have to do is to save the current value somewhere and later compare the two. For this we use `\seriesdefault@kernel`.

```
773 \expandafter\def\expandafter\seriesdefault\expandafter{\seriesdefault\empty}
774 \let\seriesdefault@kernel\seriesdefault
```

(End of definition for `\seriesdefault` and `\seriesdefault@kernel`.)

\@acci We also save the values of some accents in `\@acci`, `\@accii` and `\@acciii` so they can
\@accii be restored by a `minipage` inside a `tabbing` environment.

```
775 \let\@acci\` \let\@accii\` \let\@acciii\=
```

(End of definition for `\@acci`, `\@accii`, and `\@acciii`.)

\cal Here were the two old *<alphabet identifiers>*.
\mit

(End of definition for \cal and \mit.)

776 ⟨/2ekernel⟩

File B

fontdef.dtx

<-latexrelease> [2021/01/15 v3.0i LaTeX Kernel (<-latexrelease> font setup)]

1 Introduction

This file is used to generate the files `fonttext.ltx` (text font declarations) and `fontmath.ltx` (math font declarations), which are used during the format generation. It contains the declaration of the standard text encodings used at the site as well as a minimal subset of font shape groups that NFSS will look at to ensure that the specified encodings are valid.

The math part contains the setup for math encodings as well as the default math symbol declarations that belong to the encoding.

It is possible to change this setup (by using other fonts, or defaults) without losing the ability to process documents written at other sites. Portability in this sense means that a document will compile without errors. It does not mean, however, that identical output will be produced. For this it is necessary that the distributed setup is used at both installations.

2 Customization

You are not allowed to change this source file! If you want to change the default encodings and/or the font shape groups preloaded you should create a copy of `fonttext.ltx` under the name `fonttext.cfg` and change this copy. If L^AT_EX 2 _{ε} 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^AT_EX versions. Thus, before sending in a bug report please try your test file with a L^AT_EX 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^AT_EX installations behave in the same way.

T1	Cork T _E X text encoding
OT1	old T _E X text encoding
U	unknown encoding
OML	old T _E X math letters encoding
OMS	old T _E X math symbols encoding
OMX	old T _E 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 \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 \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 \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 \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^AT_EX 2 _{ε} 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^AT_EX 2 _{ε} 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^AT_EX 2 _{ε} 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^AT_EX 2 _{ε} 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_EX 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>/text</code>
	<code>\IncludeInRelease{2017/01/01}%</code>
	<code>\encodingdefault{TU encoding default} %</code>
	<code>\ifx\Umathcode\@undefined</code>
	<code>\renewcommand\encodingdefault{OT1}</code>
	<code>\fontencoding{\encodingdefault}</code>
	<code>\renewcommand\rmdefault{cmr}</code>
	<code>\fontfamily{\rmdefault}</code>
	<code>\renewcommand\sffamily{cmss}</code>
	<code>\renewcommand\ttdefault{cmtt}</code>
	<code>\else</code>
	<code>\renewcommand\encodingdefault{TU}</code>
	<code>\%done in every job\fontencoding{\encodingdefault}</code>
	<code>\renewcommand\rmdefault{lmr}</code>
	<code>\fontfamily{\rmdefault}</code>
	<code>\renewcommand\sffamily{lmss}</code>
	<code>\renewcommand\ttdefault{lmtt}</code>
	<code>\fi</code>
	<code>\EndIncludeInRelease</code>
	<code>\IncludeInRelease{0000/00/00}%</code>
	<code>\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 of 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 of definition for `\bfdefault` and `\mddefault`.)

`\itdefault` Shape changing commands use the following hooks.

```

94  \newcommand\itdefault{it}
95  \newcommand\sldefault{sl}
96  \newcommand\scdefault{sc}
97  \newcommand\updefault{up} % overwritten below (for rollback)

```

(End of definition for `\itdefault` and others.)

```

98  
```

- 99 {*text | latexrelease}
- 100 \let\fontencoding{\OT1}
- 101 \let\renewcommand\encodingdefault{\OT1}
- 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 
```

- 108
- 109 \let\fontencoding{\OT1}
- 110 \let\renewcommand\encodingdefault{\OT1}
- 111 \let\fontencoding{\encodingdefault}
- 112 \let\renewcommand\rmdefault{cmr}
- 113 \let\fontfamily{\rmdefault}
- 114 \let\renewcommand\sffamily{\cmss}
- 115 \let\renewcommand\ttdefault{\cmtt}
- 116 \let\renewcommand\updefault{n\@empty}
- 117 \let\renewcommand\bfdefault{bx\@empty}
- 118 \let\renewcommand\mddefault{m\@empty}
- 119 \let\renewcommand\updefault{up\@empty}

\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 of 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^AT_EX. 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^AT_EX. 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 InTeX. 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 \rm \delcode values to -1, except $\text{\rm \delcode' .}=0$

```

255 \DeclareMathDelimiter{()}{\mathopen}{operators}{28}{largesymbols}{00}
256 \DeclareMathDelimiter{}{\mathclose}{operators}{29}{largesymbols}{01}
257 \DeclareMathDelimiter{[]}{\mathopen}{operators}{5B}{largesymbols}{02}
258 \DeclareMathDelimiter{}{\mathclose}{operators}{5D}{largesymbols}{03}

```

The next two are considered to be relations when not used in the context of a delimiter! And worse, they do even represent different glyphs when being used as delimiter and not as delimiter. This is a user level syntax inherited from plain TeX. Therefore we explicitly redefine the math symbol definitions for these symbols afterwards.

```

259 \DeclareMathDelimiter{<}{\mathopen}{symbols}{68}{largesymbols}{0A}
260 \DeclareMathDelimiter{>}{\mathclose}{symbols}{69}{largesymbols}{0B}
261 \DeclareMathSymbol{<}{\mathrel}{letters}{3C}
262 \DeclareMathSymbol{>}{\mathrel}{letters}{3E}

```

And here is another case where the non-delimiter version produces a glyph different from the delimiter version.

```

263 \DeclareMathDelimiter{/}{\mathord}{operators}{2F}{largesymbols}{0E}
264 \DeclareMathSymbol{/}{\mathord}{letters}{3D}
265 \DeclareMathDelimiter{|}{\mathord}{symbols}{6A}{largesymbols}{0C}
266 \expandafter\DeclareMathDelimiter@\backslashchar
267 \mathord{symbols}{6E}{largesymbols}{0F}

```

N.B. { and } should NOT get delcodes; otherwise parameter grouping fails!

6.4 Symbols accessed via control sequences

6.4.1 Greek letters

```

268 \DeclareMathSymbol{\alpha}{\mathord}{letters}{0B}
269 \DeclareMathSymbol{\beta}{\mathord}{letters}{0C}
270 \DeclareMathSymbol{\gamma}{\mathord}{letters}{0D}
271 \DeclareMathSymbol{\delta}{\mathord}{letters}{0E}
272 \DeclareMathSymbol{\epsilon}{\mathord}{letters}{0F}
273 \DeclareMathSymbol{\zeta}{\mathord}{letters}{10}
274 \DeclareMathSymbol{\eta}{\mathord}{letters}{11}
275 \DeclareMathSymbol{\theta}{\mathord}{letters}{12}
276 \DeclareMathSymbol{\iota}{\mathord}{letters}{13}
277 \DeclareMathSymbol{\kappa}{\mathord}{letters}{14}
278 \DeclareMathSymbol{\lambda}{\mathord}{letters}{15}
279 \DeclareMathSymbol{\mu}{\mathord}{letters}{16}
280 \DeclareMathSymbol{\nu}{\mathord}{letters}{17}
281 \DeclareMathSymbol{\xi}{\mathord}{letters}{18}
282 \DeclareMathSymbol{\pi}{\mathord}{letters}{19}
283 \DeclareMathSymbol{\rho}{\mathord}{letters}{1A}
284 \DeclareMathSymbol{\sigma}{\mathord}{letters}{1B}
285 \DeclareMathSymbol{\tau}{\mathord}{letters}{1C}
286 \DeclareMathSymbol{\upsilon}{\mathord}{letters}{1D}
287 \DeclareMathSymbol{\phi}{\mathord}{letters}{1E}
288 \DeclareMathSymbol{\chi}{\mathord}{letters}{1F}
289 \DeclareMathSymbol{\psi}{\mathord}{letters}{20}
290 \DeclareMathSymbol{\omega}{\mathord}{letters}{21}
291 \DeclareMathSymbol{\varepsilon}{\mathord}{letters}{22}
292 \DeclareMathSymbol{\vartheta}{\mathord}{letters}{23}
293 \DeclareMathSymbol{\varpi}{\mathord}{letters}{24}

```

```

294 \DeclareMathSymbol{\varrho}{\mathord}{letters}{"25}
295 \DeclareMathSymbol{\varsigma}{\mathord}{letters}{"26}
296 \DeclareMathSymbol{\varphi}{\mathord}{letters}{"27}
297 \DeclareMathSymbol{\Gamma}{\mathalpha}{operators}{"00}
298 \DeclareMathSymbol{\Delta}{\mathalpha}{operators}{"01}
299 \DeclareMathSymbol{\Theta}{\mathalpha}{operators}{"02}
300 \DeclareMathSymbol{\Lambda}{\mathalpha}{operators}{"03}
301 \DeclareMathSymbol{\Xi}{\mathalpha}{operators}{"04}
302 \DeclareMathSymbol{\Pi}{\mathalpha}{operators}{"05}
303 \DeclareMathSymbol{\Sigma}{\mathalpha}{operators}{"06}
304 \DeclareMathSymbol{\Upsilon}{\mathalpha}{operators}{"07}
305 \DeclareMathSymbol{\Phi}{\mathalpha}{operators}{"08}
306 \DeclareMathSymbol{\Psi}{\mathalpha}{operators}{"09}
307 \DeclareMathSymbol{\Omega}{\mathalpha}{operators}{"0A}

```

6.4.2 Ordinary symbols

```

308 \DeclareMathSymbol{\aleph}{\mathord}{symbols}{"40}
309 \DeclareMathSymbol{\imath}{\mathord}{letters}{"7B}
310 \DeclareMathSymbol{\jmath}{\mathord}{letters}{"7C}
311 \DeclareMathSymbol{\ell}{\mathord}{letters}{"60}
312 \DeclareMathSymbol{\wp}{\mathord}{letters}{"7D}
313 \DeclareMathSymbol{\Re}{\mathord}{symbols}{"3C}
314 \DeclareMathSymbol{\Im}{\mathord}{symbols}{"3D}
315 \DeclareMathSymbol{\partial}{\mathord}{letters}{"40}
316 \DeclareMathSymbol{\infty}{\mathord}{symbols}{"31}
317 \DeclareMathSymbol{\prime}{\mathord}{symbols}{"30}
318 \DeclareMathSymbol{\emptyset}{\mathord}{symbols}{"3B}
319 \DeclareMathSymbol{\nabla}{\mathord}{symbols}{"72}
320 \DeclareMathSymbol{\top}{\mathord}{symbols}{"3E}
321 \DeclareMathSymbol{\bot}{\mathord}{symbols}{"3F}
322 \DeclareMathSymbol{\triangle}{\mathord}{symbols}{"34}
323 \DeclareMathSymbol{\forall}{\mathord}{symbols}{"38}
324 \DeclareMathSymbol{\exists}{\mathord}{symbols}{"39}
325 \DeclareMathSymbol{\neg}{\mathord}{symbols}{"3A}

```

Alias:

```

326 %     \let\lnot=\neg
327 \DeclareMathSymbol{\lnot}{\mathord}{symbols}{"3A}
328 \DeclareMathSymbol{\flat}{\mathord}{letters}{"5B}
329 \DeclareMathSymbol{\natural}{\mathord}{letters}{"5C}
330 \DeclareMathSymbol{\sharp}{\mathord}{letters}{"5D}
331 \DeclareMathSymbol{\clubsuit}{\mathord}{symbols}{"7C}
332 \DeclareMathSymbol{\diamondsuit}{\mathord}{symbols}{"7D}
333 \DeclareMathSymbol{\heartsuit}{\mathord}{symbols}{"7E}
334 \DeclareMathSymbol{\spadesuit}{\mathord}{symbols}{"7F}

335 \ DeclareRobustCommand{\hbar}{\mathchar'26\mkern-9mu h}
336 \ DeclareRobustCommand{\surd}{\mathchar"1270}
337 \ DeclareRobustCommand{\angle}{\vbox{\ialign{$\m@th\scriptstyle##$\crcr
338     \not\mathrel{\mkern14mu}\crcr
339     \noalign{\nointerlineskip}
340     \mkern2.5mu\leaders\hrule\height.34pt\hfill\mkern2.5mu\crcr}}}

```

6.4.3 Large Operators

```

341 \DeclareMathSymbol{\coprod}{\mathop}{largesymbols}{"60}

```

```

342 \DeclareMathSymbol{\bigvee}{\mathop}{largesymbols}{57}
343 \DeclareMathSymbol{\bigwedge}{\mathop}{largesymbols}{56}
344 \DeclareMathSymbol{\biguplus}{\mathop}{largesymbols}{55}
345 \DeclareMathSymbol{\bigcap}{\mathop}{largesymbols}{54}
346 \DeclareMathSymbol{\bigcup}{\mathop}{largesymbols}{53}
347 \DeclareMathSymbol{\intop}{\mathop}{largesymbols}{52}
348     \ DeclareRobustCommand\int{\intop\nolimits}
349 \DeclareMathSymbol{\prod}{\mathop}{largesymbols}{51}
350 \DeclareMathSymbol{\sum}{\mathop}{largesymbols}{50}
351 \DeclareMathSymbol{\bigotimes}{\mathop}{largesymbols}{4E}
352 \DeclareMathSymbol{\bigoplus}{\mathop}{largesymbols}{4C}
353 \DeclareMathSymbol{\bigodot}{\mathop}{largesymbols}{4A}
354 \DeclareMathSymbol{\ointop}{\mathop}{largesymbols}{48}
355     \ DeclareRobustCommand\oint{\ointop\nolimits}
356 \DeclareMathSymbol{\bigsqcup}{\mathop}{largesymbols}{46}
357 \DeclareMathSymbol{\smallint}{\mathop}{symbols}{73}

```

6.4.4 Binary symbols

```

358 \DeclareMathSymbol{\triangleleft}{\mathbin}{letters}{2F}
359 \DeclareMathSymbol{\triangleright}{\mathbin}{letters}{2E}
360 \DeclareMathSymbol{\bigtriangleup}{\mathbin}{symbols}{34}
361 \DeclareMathSymbol{\bigtriangledown}{\mathbin}{symbols}{35}

```

Alias:

```

362 %   \let \varbigtriangledown \bigtriangledown
363 %   \let \varbigtriangleup \bigtriangleup
364 \DeclareMathSymbol{\varbigtriangleup}{\mathbin}{symbols}{34}
365 \DeclareMathSymbol{\varbigtriangledown}{\mathbin}{symbols}{35}

```

These last two synonyms are needed because the `stmaryrd` package redefines them as Operators.

```

366 \DeclareMathSymbol{\wedge}{\mathbin}{symbols}{5E}
367 \DeclareMathSymbol{\vee}{\mathbin}{symbols}{5F}

```

Alias:

```

368 %   \let\land=\wedge
369 %   \let\lor=\vee
370 \DeclareMathSymbol{\land}{\mathbin}{symbols}{5E}
371 \DeclareMathSymbol{\lor}{\mathbin}{symbols}{5F}
372 \DeclareMathSymbol{\cap}{\mathbin}{symbols}{5C}
373 \DeclareMathSymbol{\cup}{\mathbin}{symbols}{5B}
374 \DeclareMathSymbol{\ddagger}{\mathbin}{symbols}{7A}
375 \DeclareMathSymbol{\dagger}{\mathbin}{symbols}{79}
376 \DeclareMathSymbol{\sqcap}{\mathbin}{symbols}{75}
377 \DeclareMathSymbol{\sqcup}{\mathbin}{symbols}{74}
378 \DeclareMathSymbol{\uplus}{\mathbin}{symbols}{5D}
379 \DeclareMathSymbol{\amalg}{\mathbin}{symbols}{71}
380 \DeclareMathSymbol{\diamond}{\mathbin}{symbols}{05}
381 \DeclareMathSymbol{\bullet}{\mathbin}{symbols}{0F}
382 \DeclareMathSymbol{\wr}{\mathbin}{symbols}{6F}
383 \DeclareMathSymbol{\div}{\mathbin}{symbols}{04}
384 \DeclareMathSymbol{\odot}{\mathbin}{symbols}{0C}
385 \DeclareMathSymbol{\oslash}{\mathbin}{symbols}{0B}
386 \DeclareMathSymbol{\otimes}{\mathbin}{symbols}{0A}
387 \DeclareMathSymbol{\ominus}{\mathbin}{symbols}{09}

```

```

388 \DeclareMathSymbol{\oplus}{\mathbin}{symbols}{08}
389 \DeclareMathSymbol{\mp}{\mathbin}{symbols}{07}
390 \DeclareMathSymbol{\pm}{\mathbin}{symbols}{06}
391 \DeclareMathSymbol{\circ}{\mathbin}{symbols}{0E}
392 \DeclareMathSymbol{\bigcirc}{\mathbin}{symbols}{0D}
393 \DeclareMathSymbol{\setminus}{\mathbin}{symbols}{6E}
394 \DeclareMathSymbol{\cdotp}{\mathbin}{symbols}{01}
395 \DeclareMathSymbol{\ast}{\mathbin}{symbols}{03}
396 \DeclareMathSymbol{\times}{\mathbin}{symbols}{02}
397 \DeclareMathSymbol{\star}{\mathbin}{letters}{3F}

```

6.4.5 Relations

```

398 \DeclareMathSymbol{\propto}{\mathrel}{symbols}{2F}
399 \DeclareMathSymbol{\sqsubseteq}{\mathrel}{symbols}{76}
400 \DeclareMathSymbol{\sqsupseteq}{\mathrel}{symbols}{77}
401 \DeclareMathSymbol{\parallel}{\mathrel}{symbols}{6B}
402 \DeclareMathSymbol{\mid}{\mathrel}{symbols}{6A}
403 \DeclareMathSymbol{\dashv}{\mathrel}{symbols}{61}
404 \DeclareMathSymbol{\vdash}{\mathrel}{symbols}{60}
405 \DeclareMathSymbol{\nearrow}{\mathrel}{symbols}{25}
406 \DeclareMathSymbol{\searrow}{\mathrel}{symbols}{26}
407 \DeclareMathSymbol{\nwarrow}{\mathrel}{symbols}{2D}
408 \DeclareMathSymbol{\swarrow}{\mathrel}{symbols}{2E}
409 \DeclareMathSymbol{\Leftrightarrow}{\mathrel}{symbols}{2C}
410 \DeclareMathSymbol{\Leftarrow}{\mathrel}{symbols}{28}
411 \DeclareMathSymbol{\Rightarrow}{\mathrel}{symbols}{29}
412 \DeclareRobustCommand{\neq}{\not=}

```

As `\neq` is robust we should not use `\let` to define `\ne` as then it would change if `\neq` changes.

```
413 \DeclareRobustCommand{\ne}{\not=}
```

It would ok to use `\let` for those declared by `\DeclareMathSymbol` but for a cleaner interface we avoid it always (just in case the internals change).

```

414 \DeclareMathSymbol{\leq}{\mathrel}{symbols}{14}
415 \DeclareMathSymbol{\geq}{\mathrel}{symbols}{15}

```

Alias:

```

416 % \let\le=\leq
417 % \let\ge=\geq
418 \DeclareMathSymbol{\le}{\mathrel}{symbols}{14}
419 \DeclareMathSymbol{\ge}{\mathrel}{symbols}{15}
420 \DeclareMathSymbol{\succ}{\mathrel}{symbols}{1F}
421 \DeclareMathSymbol{\prec}{\mathrel}{symbols}{1E}
422 \DeclareMathSymbol{\approx}{\mathrel}{symbols}{19}
423 \DeclareMathSymbol{\succeq}{\mathrel}{symbols}{17}
424 \DeclareMathSymbol{\preceq}{\mathrel}{symbols}{16}
425 \DeclareMathSymbol{\supset}{\mathrel}{symbols}{1B}
426 \DeclareMathSymbol{\subset}{\mathrel}{symbols}{1A}
427 \DeclareMathSymbol{\supseteq}{\mathrel}{symbols}{13}
428 \DeclareMathSymbol{\subseteq}{\mathrel}{symbols}{12}
429 \DeclareMathSymbol{\in}{\mathrel}{symbols}{32}
430 \DeclareMathSymbol{\ni}{\mathrel}{symbols}{33}

```

Alias:

```
431 % \let\owns=\ni
```

```

432 \DeclareMathSymbol{\owns}{\mathrel}{symbols}{33}
433 \DeclareMathSymbol{\gg}{\mathrel}{symbols}{1D}
434 \DeclareMathSymbol{\ll}{\mathrel}{symbols}{1C}
435 \DeclareMathSymbol{\not}{\mathrel}{symbols}{36}
436 \DeclareMathSymbol{\leftrightarrow}{\mathrel}{symbols}{24}
437 \DeclareMathSymbol{\leftarrow}{\mathrel}{symbols}{20}
438 \DeclareMathSymbol{\rightarrow}{\mathrel}{symbols}{21}

Alias:
439 %   \let\gets=\leftarrow
440 %   \let\to=\rightarrow
441 \DeclareMathSymbol{\gets}{\mathrel}{symbols}{20}
442 \DeclareMathSymbol{\to}{\mathrel}{symbols}{21}
443 \DeclareMathSymbol{\mapstochar}{\mathrel}{symbols}{37}
444     \ DeclareRobustCommand{\mapsto}{\mapstochar\rightarrow}
445 \DeclareMathSymbol{\sim}{\mathrel}{symbols}{18}
446 \DeclareMathSymbol{\simeq}{\mathrel}{symbols}{27}
447 \DeclareMathSymbol{\perp}{\mathrel}{symbols}{3F}
448 \DeclareMathSymbol{\equiv}{\mathrel}{symbols}{11}
449 \DeclareMathSymbol{\asymp}{\mathrel}{symbols}{10}
450 \DeclareMathSymbol{\smile}{\mathrel}{letters}{5E}
451 \DeclareMathSymbol{\frown}{\mathrel}{letters}{5F}
452 \DeclareMathSymbol{\leftharpoonup}{\mathrel}{letters}{28}
453 \DeclareMathSymbol{\leftharpoondown}{\mathrel}{letters}{29}
454 \DeclareMathSymbol{\rightharpoonup}{\mathrel}{letters}{2A}
455 \DeclareMathSymbol{\rightharpoondown}{\mathrel}{letters}{2B}

```

Here cometh much profligate robustification of math constructs. Warning: some of these commands may become non-robust if an AMS package is loaded.

Further potential problems: some math font packages may make unfortunate assumptions about some of these definitions that are not true of the robust versions we need.

```

456 \DeclareRobustCommand
457   \cong{\mathrel{\mathpalette\overeq\sim}} % congruence sign
458 \def\overeq#1#2{\lower.5\p@\vbox{\lineskip\maxdimen\lineskip-.5\p@
459   \ialign{$\m@th#1\hfil##\hfil$\crcr#2\crcr=\crcr}}}
460 \DeclareRobustCommand
461   \notin{\mathrel{\mathpalette\cncel\in}}
462 \def\cncel#1#2{\m@th\ooalign{$\hfil#1\mkern1mu/\hfil$\crcr$#1#2$}}
463 \DeclareRobustCommand
464   \rightleftharpoons{\mathrel{\mathpalette\rlh@{}}}
465 \def\rlh@#1{\vcenter{\m@th\hbox{\ooalign{\raise2pt
466   \hbox{$\#1\rightharpoonup$}\crcr
467   $\#1\leftharpoondown$}}}}
468 \DeclareRobustCommand
469   \doteq{\mathrel{\textstyle.\over=}}

```

6.4.6 Arrows

```

470 \DeclareRobustCommand
471   \joinrel{\mathrel{\mkern-3mu}}
472 \DeclareRobustCommand
473   \relbar{\mathrel{\smash-}} % \smash, because -
474                           % has the same height as +

```

In contrast to `plain.tex` `\Relbar` got braces around the equal sign to guard against it being “math active” expanding to `\futurelet...`. This might be the case when packages are implementing shorthands for math, e.g. `=` meaning `\Rightarrow` etc. It would actually be better not to use `=` in such definitions but instead define something like `\mathequalsign` and use this. However we can’t do this now as it would break other math layouts where characters are in different places (since those wouldn’t know about the need for a new command name).

```

475 \DeclareRobustCommand
476   \Relbar{\mathrel{=}}
477 \DeclareMathSymbol{\lhook}{\mathrel}{letters}{`2C}
478   \DeclareRobustCommand\hookrightarrow{\lhook\joinrel\rightarrow}
479 \DeclareMathSymbol{\rhook}{\mathrel}{letters}{`2D}
480   \DeclareRobustCommand\hookleftarrow{\leftarrow\joinrel\rhook}
481 \DeclareRobustCommand
482   \bowtie{\mathrel\triangleleft\joinrel\mathrel\triangleleft}
483 \DeclareRobustCommand
484   \models{\mathrel{!}\joinrel\Relbar}
485 \DeclareRobustCommand
486   \Longrightarrow{\Relbar\joinrel\rightarrow}

```

LaTeX Change: `\longrightarrow` and `\longleftarrow` redefined to make them robust.

```

487 \DeclareRobustCommand\longrightarrow
488   {\relbar\joinrel\rightarrow}
489 \DeclareRobustCommand\longleftarrow
490   {\leftarrow\joinrel\relbar}
491 \DeclareRobustCommand
492   \Longrightarrow{\Leftarrow\joinrel\Relbar}
493 \DeclareRobustCommand
494   \longmapsto{\mapstochar\longrightarrow}
495 \DeclareRobustCommand
496   \longleftrightarrow{\leftarrow\joinrel\rightarrow}
497 \DeclareRobustCommand
498   \Longleftrightarrow{\Leftarrow\joinrel\Rightarrow}
499 \DeclareRobustCommand
500   \iiff{;}{\Longleftrightarrow{};}

```

6.4.7 Punctuation symbols

```

501 \DeclareMathSymbol{\ldotp}{\mathpunct}{letters}{`3A}
502 \DeclareMathSymbol{\cdotp}{\mathpunct}{symbols}{`01}
503 \DeclareMathSymbol{\colon}{\mathpunct}{operators}{`3A}

```

This is commented out, since `\ldots` is now defined in `ltoutenc.dtx`.

```

504 %\def\@ldots{\mathinner{\ldotp\ldotp\ldotp}}
505 %\ DeclareRobustCommand\ldots
506 %   {\relax\ifmmode\@ldots\else\mbox{$\m@th\@ldots$}\fi}
507 \DeclareRobustCommand
508   \cdots{\mathinner{\cdotp\cdotp\cdotp}}
509 \DeclareRobustCommand
510   \vdots{\vbox{\baselineskip4\p@\lineskiplimit\z@
511   \kern6\p@\hbox{.}\hbox{.}\hbox{.}}}
512 \DeclareRobustCommand
513   \ddots{\mathinner{\mkern1mu\raise7\p@
514   \vbox{\kern7\p@\hbox{.}\hbox{.}}\mkern2mu}

```

```
515     \raise4\p@\hbox{.}\mkern2mu\raise\p@\hbox{.}\mkern1mu}}
```

6.4.8 Math accents

```
516 \DeclareMathAccent{\acute}{\mathalpha}{operators}{13}
517 \DeclareMathAccent{\grave}{\mathalpha}{operators}{12}
518 \DeclareMathAccent{\ddot}{\mathalpha}{operators}{7F}
519 \DeclareMathAccent{\tilde}{\mathalpha}{operators}{7E}
520 \DeclareMathAccent{\bar}{\mathalpha}{operators}{16}
521 \DeclareMathAccent{\breve}{\mathalpha}{operators}{15}
522 \DeclareMathAccent{\check}{\mathalpha}{operators}{14}
523 \DeclareMathAccent{\hat}{\mathalpha}{operators}{5E}
524 \DeclareMathAccent{\vec}{\mathord}{letters}{7E}
525 \DeclareMathAccent{\dot}{\mathalpha}{operators}{5F}
526 \DeclareMathAccent{\widetilde}{\mathord}{largesymbols}{65}
527 \DeclareMathAccent{\widehat}{\mathord}{largesymbols}{62}
```

For some reason plain TeX never bothered to provide a ring accent in math (although it is available in the fonts), but since we got a request for it here we go:

```
528 \DeclareMathAccent{\mathring}{\mathalpha}{operators}{17}
```

6.4.9 Radicals

```
529 \DeclareMathRadical{\sqrtsign}{symbols}{70}{largesymbols}{70}
```

6.4.10 Over and under something, etc

```
530 \DeclareRobustCommand\overrightarrow[1]{\vbox{\m@th\ialign{##\crcr
531     \rightarrowfill\crcr\noalign{\kern-\p@\nointerlineskip}
532     $ \hfil\displaystyle{#1} \hfil$ \crcr}}
533 \DeclareRobustCommand\overleftarrow[1]{\vbox{\m@th\ialign{##\crcr
534     \leftarrowfill\crcr\noalign{\kern-\p@\nointerlineskip}
535     $ \hfil\displaystyle{#1} \hfil$ \crcr}}
536 \DeclareRobustCommand\overbrace[1]
537     {\mathop{\vbox{\m@th\ialign{##\crcr\noalign{\kern3\p@}%
538         \downbracefill\crcr\noalign{\kern3\p@\nointerlineskip}%
539         $ \hfil\displaystyle{#1} \hfil$ \crcr}}}\limits}
540 \DeclareRobustCommand\underbrace[1]{\mathop{\vtop{\m@th\ialign{##\crcr
541     $ \hfil\displaystyle{#1} \hfil$ \crcr
542     \noalign{\kern3\p@\nointerlineskip}%
543     \upbracefill\crcr\noalign{\kern3\p@}}}\limits}
```

(quite a waste of tokens, IMHO — Frank)

```
544 \DeclareRobustCommand\skew[3]
545   {{\muskip\z@#1mu\divide\muskip\z@\tw@ \mkern\muskip\z@
546   #2{\mkern-\muskip\z@#3}\mkern\muskip\z@\mkern-\muskip\z@}{}}
547 \DeclareRobustCommand\rightarrowfill{$\m@th\smash-\mkern-7mu%
548 \cleaders\hbox{$\mkern-2mu\smash-\mkern-2mu$}\hfill
549 \mkern-7mu\mathord\rightarrow$}
550 \DeclareRobustCommand\leftarrowfill{$\m@th\mathord\leftarrow\mkern-7mu%
551 \cleaders\hbox{$\mkern-2mu\smash-\mkern-2mu$}\hfill
552 \mkern-7mu\smash-$}
553 \DeclareMathSymbol{\braceleft}{\mathord}{largesymbols}{7A}
554 \DeclareMathSymbol{\bracerd}{\mathord}{largesymbols}{7B}
555 \DeclareMathSymbol{\bracelu}{\mathord}{largesymbols}{7C}
556 \DeclareMathSymbol{\braceru}{\mathord}{largesymbols}{7D}
557 \DeclareRobustCommand\downbracefill{$\m@th \setbox\z@\hbox{$\braceleft$}%
558 \braceleft\leaders\vrule \height\ht\z@ \depth\z@\hfill\braceru
```

```

559   \braceleft\leaders\vrule \@height\ht\z@\@depth\z@\hfill\bracerd$}
560 \DeclareRobustCommand\upbracefill{$\mathop{\setbox\z@\hbox{$\braceleft$}}\limits^{\mathop{\setbox\z@\hbox{$\bracerd$}}\limits_{\mathop{\setbox\z@\hbox{$\bracerd$}}\limits}}$}
561   \braceleft\leaders\vrule \@height\ht\z@\@depth\z@\hfill\bracerd
562   \bracerd\leaders\vrule \@height\ht\z@\@depth\z@\hfill\braceru$}

```

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}{OD}

\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}{OD}}
576 \DeclareMathDelimiter{\vert}{%
577   {\mathord}{symbols}{6A}{largesymbols}{OC}}
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}{OF}}
592 \DeclareMathDelimiter{\rangle}{%
593   {\mathclose}{symbols}{69}{largesymbols}{OB}}
594 \DeclareMathDelimiter{\langle}{%
595   {\mathopen}{symbols}{68}{largesymbols}{OA}}
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 of definition for \lgroup, \rgroup, and \bracevert.)

6.5 Math versions of text commands

The \mathunderscore here is really a text definition, so it has been put back into `ltoutenc.dtx` (by Chris, 30/04/97) and should be removed from here.

These symbols are the math versions of text commands such as \P, \\$, etc.

\mathparagraph These math symbols are not in plain TeX.

```

\mathsection 614 \DeclareMathSymbol{\mathparagraph}{\mathord}{symbols}{7B}
\mathdollar 615 \DeclareMathSymbol{\mathsection}{\mathord}{symbols}{78}
\mathsterling 616 \DeclareMathSymbol{\mathdollar}{\mathord}{operators}{24}
\mathunderscore 617 \ DeclareRobustCommand{\mathsterling}{\mathit{\mathchar"7024}}
618 \ DeclareRobustCommand{\mathunderscore}{\kern.06em\vbox{\hrule\@width.3em}}
```

(End of definition for \mathparagraph and others.)

\mathellipsis This is plain TeX's \ldots.

```
619 \ DeclareRobustCommand{\mathellipsis}{\mathinner{\ldotp\ldotp\ldotp}}%
```

(End of definition for \mathellipsis.)

6.6 Other special functions and parameters

6.6.1 Biggggg

```

620 </math>
621 <math | latexrelease>
622 <mathrelease>\IncludeInRelease{2018/12/01}%
623 <mathrelease> {\Big}{Start LR-mode}%
624 \DeclareRobustCommand{\big[1]{\leavevmode@ifvmode
625   {\hbox{$\left.\#1\right.\vbox{to8.5\p@{}\right.\n@space$}}}}
626 \DeclareRobustCommand{\Big[1]{\leavevmode@ifvmode
627   {\hbox{$\left.\#1\right.\vbox{to11.5\p@{}\right.\n@space$}}}}
628 \DeclareRobustCommand{\bigg[1]{\leavevmode@ifvmode
629   {\hbox{$\left.\#1\right.\vbox{to14.5\p@{}\right.\n@space$}}}}
630 \DeclareRobustCommand{\Bigg[1]{\leavevmode@ifvmode
631   {\hbox{$\left.\#1\right.\vbox{to17.5\p@{}\right.\n@space$}}}}
632 </math | latexrelease>
```

```

633  \end{macro}
634  \end{macro}
635  \end{macro}
636  \def\big#1{\hbox{$\left#1\vbox{to8.5\p@{}\right.\n@space$}$}}
637  \def\Big#1{\hbox{$\left#1\vbox{to11.5\p@{}\right.\n@space$}$}}
638  \def\bigg#1{\hbox{$\left#1\vbox{to14.5\p@{}\right.\n@space$}$}}
639  \def\Bigg#1{\hbox{$\left#1\vbox{to17.5\p@{}\right.\n@space$}$}}
640  \end{macro}
641  \end{macro}
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 of 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  \end{math}

```

7 Default cfg files

We provide default cfg files here to ensure that on installations that search large file trees we do not pick up some strange customisation files from somewhere.

```

648  <*cfgtext | cfgmath | cfgprel>
649  %%
650  %%
651  %%
652  %% Load the standard setup:
653  %%
654  <+cfgtext> \input{fonttext.ltx}
655  <+cfgmath> \input{fontmath.ltx}
656  <+cfgprel> \input{preload.ltx}
657  %%
658  %% Small changes could go here; see documentation in cfgguide.tex for
659  %% allowed modifications.
660  %%
661  %% In particular it is not allowed to misuse this configuration file
662  %% to modify internal LaTeX commands!
663  %%
664  %% If you use this file as the basis for configuration please change
665  %% the \ProvidesFile lines to clearly identify your modification, e.g.,
666  %%
667  <+cfgtext>%> \ProvidesFile{fonttext.cfg}[2001/06/01
668  <+cfgmath>%> \ProvidesFile{fonttext.cfg}[2001/06/01
669  <+cfgprel>%> \ProvidesFile{preload.cfg}[2001/06/01
670  %%                                         Customised local font setup]
671  %%

```

672 %%
673 ⟨/cfgtext | cfgmath | cfgprel⟩

File C

preload.dtx

1 Overview

This file contains a number of possible settings for preloading fonts during installation of NFSS2 (which is used by L^AT_EX 2 _{ε}). 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^AT_EX 2 _{ε} 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^AT_EX 2 _{ε} 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^AT_EX 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 \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 \LaTeX line and circle fonts. It is necessary to do this explicitly since these are used by the \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 D

ltfntcmd.dtx

Abstract

The commands defined in this file `ltfntcmd` are part of the kernel code for L^AT_EX 2_&/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_EX system and for several reasons it is better to avoid them on the user level whenever possible. In L^AT_EX3 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`.³⁴ 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
```

³⁴Any 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 bold 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^AT_EX 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^AT_EX'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 of definition for `\DeclareTextFontCommand`.)

`\textrm` Now we define the `\text<family>` commands in terms of the above; `\textttt` does not look very nice!

`\textsf` 15 `\DeclareTextFontCommand{\textrm}{\rmfamily}`
`\textnormal` 16 `\DeclareTextFontCommand{\textsf}{\sffamily}`
17 `\DeclareTextFontCommand{\textttt}{\ttfamily}`
18 `\DeclareTextFontCommand{\textnormal}{\normalfont}`

(End of definition for `\textrm` and others.)

`\textbf` For the series attribute:

`\textmd` 19 `\DeclareTextFontCommand{\textbf}{\bfseries}`
20 `\DeclareTextFontCommand{\textmd}{\mdseries}`

(End of 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 of 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 `{*2ekernel}`

(End of definition for `textulc`, `textsw`, and `textssc`.)

`\emph` Finally we have the `\em` font change declaration of L^AT_EX. The corresponding definition with argument is

42 `\DeclareTextFontCommand{\emph}{\em}`

(End of definition for `\emph`.)

`\nocorr` This is just a label, so it does nothing; it should also be unexpandable.

43 `\let\nocorr\relax`

(End of 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 of 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^AT_EX or actually already in plain T_EX. 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 of 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 of 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     \nocorrlist
```

We have to hide the `\@let@token` in the macro `\t@st@ic` rather than testing it directly in the loop since it might be `\let` to a `\fi` or `\else`, which would result in chaos.

```

90 \do \t@st@ic
```

Frank thinks that the next bit is inefficient if done after the second change. Chris thinks that most all of this is inefficient for the commonest cases: but that is the price of a cleverer algorithm. It is certainly needed to deal with the use of `\nolinebreak`.

```

91     \ifmaybe@ic \sw@slant \fi
92     \fi
93 }
```

(End of 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 of 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^AT_EX 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 of 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 of definition for `\nocorrlist`.)

- `\nfss@text` This command will by default behave like a L^AT_EX `\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 of 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}{\mathsf}
\DeclareOldFontCommand{\tt}{\normalfont\ttfamily}{\mathtt}

125 \def \DeclareOldFontCommand #1#2#3{%
126   \ DeclareRobustCommand #1{\@fontswitch {#2}{#3}}%
127 }

```

(End of 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 of 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 of definition for \normalsize.)
```

File E

lttextcomp.dtx

This file contains the implementation for accessing the glyphs provided by the TS1 encoding (Text Companion Encoding). This is now offered as part of the kernel and so the `textcomp` package which used to provide the definitions is now mainly needed for compatibility reasons (and doesn't do much any more).

```
1  {*2ekernel | latexrelease}
2  <latexrelease>\NewModuleRelease{2020/02/02}{lttextcomp}
3  <latexrelease>          {Text Companion symbols}
```

`\oldstylenums` 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^AT_EX 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\sff@ult
30         \fontfamily\sfsbstdefault
31     \else\ifx\f@family\ttdef@ult
32         \fontfamily\ttsubstdefault
33     \else
34         \fontfamily\textcompsubstdefault
35         \fi\fi\fi
36         \fontencoding{TS1}\selectfont#1%
37     \egroup
38 }
```

(End of definition for `\oldstylenums` and `\legacyoldstylenums`.)

`\textcompsubstdefault` Here is the default for the “unknown” case:

```
39 \def\textcompsubstdefault{\rmsubstdefault}
```

(End of 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}}%
```

This declaration should be usable in `.fd` files and therefore has to make its definition globally, because such files can get loaded in random places.

```
57   \global\@namedef{#1:#2}{#3}%
58   Any reason to allow those in the middle of documents?
59   \onlypreamble\DeclareEncodingSubset
60   \onlypreamble\DeclareEncodingSubset@aux
61   \onlypreamble\@DeclareEncodingSubset
```

(End of definition for `\DeclareEncodingSubset`.)

`\CheckEncodingSubset` The command `\CheckEncodingSubset` will check if the current font family has the right encoding subset to typeset a certain command. It takes five arguments as follows: first argument is either `\UseTextSymbol`, `\UseTextAccent` depending on whether or not the symbol is a text symbol or a text accent.

The second argument is the encoding from which this symbol should be fetched.

The third argument is either a fake accessor command or an error message. the code in that argument (if ever executed) receives two arguments: #2 and #5 of `\CheckEncodingSubset`.

Argument four is the subset encoding id to test against: if this value is higher than the subset id of the current font family then we typeset the symbol, i.e., execute #1{#2}#5 otherwise it runs #3#5, e.g., to produce an error message or fake the glyph somehow.

Argument five is the symbol or accent command that is being checked.

For usage examples see definitions below.

```
61 \def\CheckEncodingSubset#1#2#3#4#5{%
62   \ifnum #4>%
63     \expandafter\ifx\csname #2:\f@family\endcsname\relax
64       \csname #2:?\endcsname
65     \else
66       \csname #2:\f@family\endcsname
67     \fi
68   \relax
69   \expandafter\@firstoftwo
70 \else
71   \expandafter\@secondoftwo
72 \fi
```

```

73   {#1{#2}}{#3}%
74   #5%
75 }

```

(End of 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{\@latex@info{#1}}
```

(End of 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 of 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 of 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                                     {TS1}\{\tc@swap@accent#1\}}
113 \def\tc@swap@accent#1#2{\UseTextAccent{T1}#1}
```

(End of 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{\textthreequartersmdash}{TS1} % subst in sub-enc 9 above
146 \DeclareTextSymbolDefault{\textthreequarters}{TS1}
147 %%\DeclareTextSymbolDefault{\textthreesuperior}{TS1} % subst in sub-enc 9 above
148 \DeclareTextSymbolDefault{\textttimes}{TS1}
149 \DeclareTextSymbolDefault{\texttrademark}{TS1}
150 %%\DeclareTextSymbolDefault{\texttwelveudash}{TS1} % subst in sub-enc 9 above
151 %%\DeclareTextSymbolDefault{\texttwosuperior}{TS1} % subst in sub-enc 9 above
152 \DeclareTextSymbolDefault{\textyen}{TS1}
153 \DeclareTextSymbolDefault{\textcapitalcompwordmark}{TS1}
154 \DeclareTextSymbolDefault{\textascendercompwordmark}{TS1}

```

In the following sections the remaining default definitions are ordered by sub-encoding in which they are become **unavailable**, i.e., they are not provided in the sub-encoding with that number and all sub-encodings with higher numbers.

Thus the symbols that are available in sub-encoding x are the symbols above (always available) and the symbols listed as becoming unavailable in sub-encodings $x + 1$ and higher.

1.1 Unavailable in sub-encoding 1 and higher (drop symbols not working in Latin Modern)

The `\textcircled` is available but the glyph is simply too small so we keep using the OMS glyph.

```

155 \DeclareTextCommandDefault{\textcircled}
156   {\CheckEncodingSubset\UseTextAccent{TS1}{\UseTextAccent{OMS}}1\textcircled}

```

1.2 Unavailable in sub-encoding 2 (majority of new OTF fonts via autoinst) and higher

```

157 \DeclareTextCommandDefault{\t}
158   {\CheckEncodingSubset\UseTextAccent{TS1}{\UseTextAccent{OML}}2\t}

```

Capital accents are really only very seldom implemented, so from sub-encoding 2 onwards we use the normal T1 accents if they are asked for in the document.

In Unicode engines we don't implement them at all but always use the basic accents instead. whether that works or not really depends on the font, something like `\"X` usually comes out wrong in Unicode engines.

```

159 \ifx\Umathcode\@undefined
160   \DeclareTextCommandDefault{\capitalacute}{\@tc@check@accent{'2}\capitalacute}
161   \DeclareTextCommandDefault{\capitalbreve}{\@tc@check@accent{\u}2\capitalbreve}
162   \DeclareTextCommandDefault{\capitalcaron}{\@tc@check@accent{\v}2\capitalcaron}
163   \DeclareTextCommandDefault{\capitalcedilla}{\@tc@check@accent{\c}2\capitalcedilla}
164   \DeclareTextCommandDefault{\capitalcircumflex}{\@tc@check@accent{\^}2\capitalcircumflex}
165   \DeclareTextCommandDefault{\capitaldieresis}{\@tc@check@accent{"2}\capitaldieresis}
166   \DeclareTextCommandDefault{\capitaldotaccent}{\@tc@check@accent{\.2}\capitaldotaccent}
167   \DeclareTextCommandDefault{\capitalgrave}{\@tc@check@accent{\`2}\capitalgrave}
168   \DeclareTextCommandDefault{\capitalhungarumlaut}{\@tc@check@accent{\H}2\capitalhungarumlaut}
169   \DeclareTextCommandDefault{\capitalmacron}{\@tc@check@accent{\=2}\capitalmacron}
170   \DeclareTextCommandDefault{\capitalogonek}{\@tc@check@accent{\k}2\capitalogonek}
171   \DeclareTextCommandDefault{\capitalring}{\@tc@check@accent{\r}2\capitalring}
172   \DeclareTextCommandDefault{\capitaltie}{\@tc@check@accent{\t}2\capitaltie}
173   \DeclareTextCommandDefault{\capitaltilde}{\@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}{\@tc@check@accent{\t}2\newtie}
189 \DeclareTextCommandDefault{\capitalnewtie}{\@tc@check@accent{\t}2\capitalnewtie}
190
191

```

In Unicode engines we just execute the simple accents:

```

192 \else
193   \DeclareTextCommandDefault{\capitalacute}{\@tabacckludge'}
194   \DeclareTextCommandDefault{\capitalbreve}{\u}
195   \DeclareTextCommandDefault{\capitalcaron}{\v}
196   \DeclareTextCommandDefault{\capitalcedilla}{\c}
197   \DeclareTextCommandDefault{\capitalcircumflex}{\^}
198   \DeclareTextCommandDefault{\capitaldieresis}{"}
199   \DeclareTextCommandDefault{\capitaldotaccent}{\.}
200   \DeclareTextCommandDefault{\capitalgrave}{\@tabacckludge'}
201   \DeclareTextCommandDefault{\capitalhungarumlaut}{\H}
202   \DeclareTextCommandDefault{\capitalmacron}{\@tabacckludge=}
203   \DeclareTextCommandDefault{\capitalogonek}{\t}
204   \DeclareTextCommandDefault{\capitalring}{\k}
205   \DeclareTextCommandDefault{\capitaltie}{\r}
206   \DeclareTextCommandDefault{\capitaltilde}{\~}
207   \DeclareTextCommandDefault{\newtie}{\t}
208

```

```
209 \fi
```

The next two symbols exist in some fonts (faked?), but we ignore that to keep the subsets reasonable compact and most important linear.

```
210 \DeclareTextCommandDefault{\textlbrackdbl}{\tc@check@symbol2\textlbrackdbl}
211 \DeclareTextCommandDefault{\textrbrackdbl}{\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{\texteightoldstyle}{\tc@check@symbol2\texteightoldstyle}
215 \DeclareTextCommandDefault{\textfiveoldstyle}{\tc@check@symbol2\textfiveoldstyle}
216 \DeclareTextCommandDefault{\textfouroldstyle}{\tc@check@symbol2\textfouroldstyle}
217 \DeclareTextCommandDefault{\textnineoldstyle}{\tc@check@symbol2\textnineoldstyle}
218 \DeclareTextCommandDefault{\textoneoldstyle}{\tc@check@symbol2\textoneoldstyle}
219 \DeclareTextCommandDefault{\textsevenoldstyle}{\tc@check@symbol2\textsevenoldstyle}
220 \DeclareTextCommandDefault{\textsixoldstyle}{\tc@check@symbol2\textsixoldstyle}
221 \DeclareTextCommandDefault{\textthreeoldstyle}{\tc@check@symbol2\textthreeoldstyle}
222 \DeclareTextCommandDefault{\texttwooldstyle}{\tc@check@symbol2\texttwooldstyle}
223 \DeclareTextCommandDefault{\textzerooldstyle}{\tc@check@symbol2\textzerooldstyle}
```

The next set of glyphs is special to $\text{T}_{\text{E}}\text{X}$ 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}{\tc@check@symbol2\textacutedbl}
235 \DeclareTextCommandDefault{\textasciacute}{\tc@check@symbol2\textasciacute}
236 \DeclareTextCommandDefault{\textasciibreve}{\tc@check@symbol2\textasciibreve}
237 \DeclareTextCommandDefault{\textasciicaron}{\tc@check@symbol2\textasciicaron}
238 \DeclareTextCommandDefault{\textasciidieresis}{\tc@check@symbol2\textasciidieresis}
239 \DeclareTextCommandDefault{\textasciigrave}{\tc@check@symbol2\textasciigrave}
240 \DeclareTextCommandDefault{\textasciimacron}{\tc@check@symbol2\textasciimacron}
241 \DeclareTextCommandDefault{\textgravedbl}{\tc@check@symbol2\textgravedbl}
242 \DeclareTextCommandDefault{\texttildelow}{\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 $\text{T}_{\text{E}}\text{X}$ 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   {\tc@check@symbol2{textcentoldstyle}}
260 \DeclareTextCommandDefault{\textcircledP}
261   {\tc@check@symbol2{textcircledP}}
262 \DeclareTextCommandDefault{\textcopyleft}
263   {\tc@check@symbol2{textcopyleft}}
264 \DeclareTextCommandDefault{\textdblhyphenchar}
265   {\tc@check@symbol2{textdblhyphenchar}}
266 \DeclareTextCommandDefault{\textdblhyphen}
267   {\tc@check@symbol2{textdblhyphen}}
268 \DeclareTextCommandDefault{\textdied}
269   {\tc@check@symbol2{textdied}}
270 \DeclareTextCommandDefault{\textdiscount}
271   {\tc@check@symbol2{textdiscount}}
272 \DeclareTextCommandDefault{\textdivorced}
273   {\tc@check@symbol2{textdivorced}}
274 \DeclareTextCommandDefault{\textdollaroldstyle}
275   {\tc@check@symbol2{textdollaroldstyle}}
276 \DeclareTextCommandDefault{\textguarani}
277   {\tc@check@symbol2{textguarani}}
278 \DeclareTextCommandDefault{\textleaf}
279   {\tc@check@symbol2{textleaf}}
280 \DeclareTextCommandDefault{\textlquill}
281   {\tc@check@symbol2{textlquill}}
282 \DeclareTextCommandDefault{\textmarried}
283   {\tc@check@symbol2{textmarried}}
284 \DeclareTextCommandDefault{\textmho}
285   {\tc@check@symbol2{textmho}}
286 \DeclareTextCommandDefault{\textmusicalnote}
287   {\tc@check@symbol2{textmusicalnote}}
288 \DeclareTextCommandDefault{\textnaira}
289   {\tc@check@symbol2{textnaira}}
290 \DeclareTextCommandDefault{\textopenbullet}
291   {\tc@check@symbol2{textopenbullet}}
292 \DeclareTextCommandDefault{\textpeso}
293   {\tc@check@symbol2{textpeso}}
294 \DeclareTextCommandDefault{\textpilcrow}
295   {\tc@check@symbol2{textpilcrow}}
296 \DeclareTextCommandDefault{\textrecipe}
297   {\tc@check@symbol2{textrecipe}}
298 \DeclareTextCommandDefault{\textreferencemark}
299   {\tc@check@symbol2{textreferencemark}}
300 \DeclareTextCommandDefault{\textrquill}
301   {\tc@check@symbol2{textrquill}}
302 \DeclareTextCommandDefault{\textservicemark}
303   {\tc@check@symbol2{textservicemark}}
304 \DeclareTextCommandDefault{\textsurd}
305   {\tc@check@symbol2{textsurd}}

```

The `\textpertenthousand` also belongs in this group but here we have a choice: in T1 there is a definition for `\textpertenthousand` making the symbol up from % and `\char 24` (twice) but in many fonts that char doesn't exist and the slot is reused for random ligatures. So better not use it because often it is wrong. But pointing to TS1 is also not great as only a few fonts have it as a real symbol, so we get a substitution to CM or LM.

Alternatively we could just state that the symbol is unavailable in those fonts. For now I substitute.

```
306 \DeclareTextCommandDefault{\textpertenthousand}
307           {\tc@check@symbol2\textpertenthousand}
308 \UndeclareTextCommand{\textpertenthousand}{T1}
```

1.3 Unavailable in sub-encoding 3 and higher

Sub-encoding 2 is the one where we loose many symbols. In the higher-numbered sub-encodings we see only a few dropped additionally.

```
309 \DeclareTextCommandDefault{\textlangle}
310           {\tc@check@symbol3\textlangle}
311 \DeclareTextCommandDefault{\textrangle}
312           {\tc@check@symbol3\textrangle}
```

1.4 Unavailable in sub-encoding 4 and higher

```
313 \DeclareTextCommandDefault{\textcolonmonetary}
314           {\tc@check@symbol4\textcolonmonetary}
315 \DeclareTextCommandDefault{\textdong}
316           {\tc@check@symbol4\textdong}
317 \DeclareTextCommandDefault{\textdownarrow}
318           {\tc@check@symbol4\textdownarrow}
319 \DeclareTextCommandDefault{\textleftarrow}
320           {\tc@check@symbol4\textleftarrow}
321 \DeclareTextCommandDefault{\textlira}
322           {\tc@check@symbol4\textlira}
323 \DeclareTextCommandDefault{\textrightarrow}
324           {\tc@check@symbol4\textrightarrow}
325 \DeclareTextCommandDefault{\textuparrow}
326           {\tc@check@symbol4\textuparrow}
327 \DeclareTextCommandDefault{\textwon}
328           {\tc@check@symbol4\textwon}
```

1.5 Unavailable in sub-encoding 5 (most older PS fonts) and higher

Most older PS fonts (supported in TeX since the early nineties when virtual fonts became available) are sorted under this sub-encoding. But in reality, many of them don't have all glyphs that should be available in sub-encoding 5. Instead they show little squares, i.e., they produce "tofu" if you are unlucky.

But the coverage is so random that it is impossible to sort them properly and if we tried to ensure that they only typeset those glyphs that are really always available, we would have to put them all into sub-encoding 9; so putting them into 5 is really a compromise.

Modern fonts usually don't typeset a tofu character if a glyph is missing. They are therefore only classified as sub-encoding 5 if they really support its glyph set completely.

```

329 \DeclareTextCommandDefault{\textestimated}
330           {\text@check@symbol5\textestimated}
331 \DeclareTextCommandDefault{\textnumero}
332           {\text@check@symbol5\textnumero}

```

1.6 Unavailable in sub-encoding 6 and higher

```

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

```

1.7 Unavailable in sub-encoding 7 and higher

```

337 \DeclareTextCommandDefault{\textfractionsolidus}
338           {\text@check@symbol7\textfractionsolidus}
339 \DeclareTextCommandDefault{\textohm}
340           {\text@check@symbol7\textohm}
341 \DeclareTextCommandDefault{\textmu}
342           {\text@check@symbol7\textmu}
343 \DeclareTextCommandDefault{\textminus}
344           {\text@check@symbol7\textminus}

```

1.8 Unavailable in sub-encoding 8 and higher

```

345 \DeclareTextCommandDefault{\textblank}
346           {\text@check@symbol8\textblank}
347 \DeclareTextCommandDefault{\textinterrobangdown}
348           {\text@check@symbol8\textinterrobangdown}
349 \DeclareTextCommandDefault{\textinterrobang}
350           {\text@check@symbol8\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}\text@fake@euro{8}\texteuro}

```

1.9 Unavailable in Sub-encoding 9 (most missing)

```

353 \DeclareTextCommandDefault{\textcelsius}
354           {\text@check@symbol9\textcelsius}
355 \DeclareTextCommandDefault{\textonesuperior}
356           {\text@check@symbol9\textonesuperior}
357 \DeclareTextCommandDefault{\textthreequartersemdash}
358           {\text@check@symbol9\textthreequartersemdash}
359 \DeclareTextCommandDefault{\textthreesuperior}
360           {\text@check@symbol9\textthreesuperior}
361 \DeclareTextCommandDefault{\texttwelveudash}
362           {\text@check@symbol9\texttwelveudash}
363 \DeclareTextCommandDefault{\texttwosuperior}
364           {\text@check@symbol9\texttwosuperior}
365 \DeclareTextCommandDefault{\textbardbl}
366           {\text@check@symbol9\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 \%DeclarerTextSymbol{\textcopyleft}{TS1}{171}
369 \%DeclarerTextSymbol{\textdblhyphen}{TS1}{45}
370 \%DeclarerTextSymbol{\textdblhyphenchar}{TS1}{127}
371 \%DeclarerTextSymbol{\textquotestraightbase}{TS1}{13}
372 \%DeclarerTextSymbol{\textquotestraightdblbase}{TS1}{18}
373 \%DeclarerTextSymbol{\textleaf}{TS1}{108}
374 \%DeclarerTextSymbol{\texttwelveudash}{TS1}{21}
375 \%DeclarerTextSymbol{\textthreequartersdash}{TS1}{22}
```

If oldstyle numerals are asked for we just use \oldstylenums.

```
376 \DeclarerTextCommand{\textzerooldstyle} \UnicodeEncodingName{\oldstylenums{0}}
377 \DeclarerTextCommand{\textoneoldstyle} \UnicodeEncodingName{\oldstylenums{1}}
378 \DeclarerTextCommand{\texttwooldstyle} \UnicodeEncodingName{\oldstylenums{2}}
379 \DeclarerTextCommand{\textthreeoldstyle} \UnicodeEncodingName{\oldstylenums{3}}
380 \DeclarerTextCommand{\textfouroldstyle} \UnicodeEncodingName{\oldstylenums{4}}
381 \DeclarerTextCommand{\textfiveoldstyle} \UnicodeEncodingName{\oldstylenums{5}}
382 \DeclarerTextCommand{\textsixoldstyle} \UnicodeEncodingName{\oldstylenums{6}}
383 \DeclarerTextCommand{\textsevenoldstyle} \UnicodeEncodingName{\oldstylenums{7}}
384 \DeclarerTextCommand{\texteightoldstyle} \UnicodeEncodingName{\oldstylenums{8}}
385 \DeclarerTextCommand{\textnineoldstyle} \UnicodeEncodingName{\oldstylenums{9}}
```

These have Unicode slots so this should be integrated into TU explicitly

```
386 \DeclarerTextSymbol{\textpilcrow} \UnicodeEncodingName{"00B6}
387 \DeclarerTextSymbol{\textborn} \UnicodeEncodingName{"002A}
388 \DeclarerTextSymbol{\textdied} \UnicodeEncodingName{"2020}
389 \DeclarerTextSymbol{\textlbrackdbl} \UnicodeEncodingName{"27E6}
390 \DeclarerTextSymbol{\textrbrackdbl} \UnicodeEncodingName{"27E7}
391 \DeclarerTextSymbol{\textguarani} \UnicodeEncodingName{"20B2}
```

We could make \textcentoldstyle and \textdollaroldstyle point to dollar and cent in the Unicode encoding

```
392 \%DeclarerTextSymbol{\textcentoldstyle} \UnicodeEncodingName{"00A2}
393 \%DeclarerTextSymbol{\textdollaroldstyle} \UnicodeEncodingName{"0024}
```

but I think it is better to pick them up from TS1 even if that usually means LMR fonts

```
394 \DeclarerTextSymbol{\textdollaroldstyle}{TS1}{138}
395 \DeclarerTextSymbol{\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 T_EX 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}{qpl}      {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}{u18}      {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}{u19}      {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}{ugg}   {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^AT_EX. 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}{fbb-*}      {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}{AccanthisADFStdNoThree-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}{GilliusADFNoTwo-LF} {5}
535 \DeclareEncodingSubset{TS1}{GilliusADFNoTwoCond-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}{LiberutilusMono-TLF} {6}
550 \DeclareEncodingSubset{TS1}{LiberutilusSerifDisplay-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 of definition for `\UseLegacyTextSymbols.`)

```

\textlegacyasteriskcentered
  \textlegacybardbl
  \textlegacybullet
\textlegacydaggerdbl
  \textlegacydagger
\textlegacyparagraph
\textlegacyperiodcentered
  \textlegacysection

```

Here are new names for the legacy symbols that L^AT_EX 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 of definition for `\textlegacyasteriskcentered` and others.)

Supporting rollback ...

```

605 </2ekernel | latexrelease>
606 <latexrelease>
607 <latexrelease>\IncludeInRelease{0000/00/00}%
608 <latexrelease>    {lttextcomp}{Undefine text companion symbols}%
609 <latexrelease>
610 <latexrelease>\DeclareRobustCommand\oldstylenums[1]{%
611 <latexrelease>    \begingroup
612 <latexrelease>    \spaceskip\fontdimen\tw@\font
613 <latexrelease>    \usefont{OML}{\rmdefault}{\f@series}{it}%
614 <latexrelease>    \mathgroup\symletters #1%
615 <latexrelease>    \endgroup
616 <latexrelease>}
617 <latexrelease>\let\legacyoldstylenums\@undefined
618 <latexrelease>\def\textcompsubstdefault{cmr}
619 <latexrelease>
620 <latexrelease>\let\DeclareEncodingSubset\@undefined
621 <latexrelease>\let\CheckEncodingSubset\@undefined
622 <latexrelease>
623 <latexrelease>\DeclareTextSymbolDefault{\textdollar}{OT1}
624 <latexrelease>\DeclareTextSymbolDefault{\textsterling}{OT1}
625 <latexrelease>\DeclareTextCommand{\textdollar}{OT1}{\hmode@bgroup
626 <latexrelease>    \ifdim \fontdimen@ne\font >\z@

```

```

627 〈latexrelease〉      \slshape
628 〈latexrelease〉      \else
629 〈latexrelease〉      \upshape
630 〈latexrelease〉      \fi
631 〈latexrelease〉      \char`\'$\\egroup}
632 〈latexrelease〉\DeclareTextCommand{\textsterling}{OT1}{\hmode@bgroup
633 〈latexrelease〉    \ifdim \fontdimen1ne\font >\z@
634 〈latexrelease〉      \itshape
635 〈latexrelease〉      \else
636 〈latexrelease〉      \fontshape{ui}\selectfont
637 〈latexrelease〉      \fi
638 〈latexrelease〉      \char`\'$\\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〉                  {\textsuperscript{a}}
664 〈latexrelease〉\DeclareTextCommandDefault{\textordmasculine}
665 〈latexrelease〉                  {\textsuperscript{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〉                  {\textsuperscript{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\textasciiacute\@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\textlangl\@undefined
763 〈latexrelease〉\let\textrangle\@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\textfractionssolidus\@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  \let\texteuro\@undefined
790  \let\textcelsius\@undefined
791  \let\textonesuperior\@undefined
792  \let\textthreequartersemdash\@undefined
793  \let\textthreesuperior\@undefined
794  \let\texttwelveudash\@undefined
795  \let\texttwosuperior\@undefined
796  \let\textbardbl\@undefined
797  \let\textlegacy\@undefined
798  \let\UseLegacyTextSymbols\@undefined
799  \let\textlegacyasteriskcentered\@undefined
800  \let\textlegacybardbl\@undefined
801  \let\textlegacybullet\@undefined
802  \let\textlegacydaggerdbl\@undefined
803  \let\textlegacydagger\@undefined
804  \let\textlegacyparagraph\@undefined
805  \let\textlegacyperiodcentered\@undefined
806  \let\textlegacysection\@undefined
807  \let\textlegacy\@undefined
808  \let\EndModuleRelease\@undefined

```

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}
825         {\gdef\tc@errorwarn{\PackageError{textcomp}{}}}
826 \DeclareOption{warn}
827         {\gdef\tc@errorwarn{\PackageWarning{textcomp}{#1}}}
828 \DeclareOption{info}

```

```

829          {\gdef\tc@errorwarn#1#2{\PackageInfo{textcomp}{#1}}}
830 \DeclareOption{quiet}{\gdef\tc@errorwarn#1#2{}}

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^AT_EX kernel.

```

845 \InputIfFileExists{textcomp.cfg}
846   {\PackageInfo{textcomp}{Local configuration file used}}{}
847 
```

5.1 The old textcomp package code

This section contains the old code for the textcomp package and its documentation. It is only used if we roll back prior to 2020. Thus all the rest is mainly for historians. Note that the old code categorized in the sub-encodings only into 6 classes not 10.

```

848 <*TS1oldsty>
849 \ProvidesPackage{textcomp}
850   [2018/08/11 v2.0j Standard LATEX package]

```

This one is for the TS1 encoding which contains text symbols for use with the T1-encoded text fonts. It therefore first inputs the file `TS1enc.def` and then sets (or resets) the defaults for the symbols it contains. The result of this is that when one of these symbols is accessed and the current encoding does not provide it, the symbol will be supplied by a silent, local change to this encoding.

Since many PostScript fonts only implement a subset of TS1 many commands only produce black blobs of ink. To resolve the resulting problems a number of options have been introduced and some code has been developed to distinguish sub-encodings.

The sub-encodings have a numerical id and are defined as follows for TS1:

#5 those TS1 symbols that are also in the ISO-Adobe character set; without `textcurrency`, which is often misused for the Euro. Older Type1 fonts from the non-T_EX world provide only this subset.

#4 = #5 + `\texteuro`. Most newer fonts provide this.

#3 = #4 + `\textomega`. Can also be described as $TS1 \cap (ISO\text{-}Adobe \cup MacRoman)$. (Except for the missing "currency".)

#2 = #3 + \textestimated + \textcurrency. Can also be described as TS1 ∩ 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., **cmt**), and the subset encoding id as a third, (e.g., 0 for **cmt**).

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 of 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 of 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     0\csname #2:?\endcsname
886     \relax
887     \expandafter\@firstoftwo
888   \else
889     \expandafter\@secondoftwo
890   \fi
891   {#1{#2}}{#3}%
892 }
```

```

892     #5%
893 }

In normal circumstances the test is a bit more complicated: first check if there exists
a macro \langle arg2\rangle:\langle current-family\rangle and if so use that value to test against, otherwise use
the default to test against.

894 \else
895 \def\CheckEncodingSubset#1#2#3#4#5{%
896   \ifnum #4>%
897     \expandafter\ifx\csname #2:\f@family\endcsname\relax
898       \csname #2:?\endcsname
899     \else
900       \csname #2:\f@family\endcsname
901     \fi
902   \relax
903   \expandafter\@firstoftwo
904 \else
905   \expandafter\@secondoftwo
906 \fi
907 {#1{#2}}{#3}%
908 #5%
909 }
910 \fi

```

(End of definition for \CheckEncodingSubset.)

```
\tc@subst
911 \def\tc@subst#1{%
912   \tc@errorwarn{textcomp}%
913   {Symbol \string#1 not provided by\MessageBreak
914     font family \f@family\space
915     in TS1 encoding.\MessageBreak Default family used instead}\@eha
916 \bgroup\fontfamily{textcomp}\substdefault\selectfont#1\egroup
917 }
```

(End of definition for \tc@subst.)

\tc@error \tc@error is going to be used in arg #3 of \CheckEncodingSubset when a symbol is not
available in a certain font family. It gets pass the encoding it normally lives in (arg one)
and the name of the symbol or accent that has a problem.

```

918 % error commands take argument:
919 % #1 symbol to be used
920 \def\tc@error#1{%
921   \PackageError{textcomp}%
922   {Accent \string#1 not provided by\MessageBreak
923     font family \f@family\space
924     in TS1 encoding}\@eha
925 }
```

(End of definition for \tc@error.)

\tc@fake@euro \tc@fake@euro is an example of a “fake” definition to use in arg #3 of \CheckEncodingSubset
when a symbol is not available in a certain font family. Here we produce an Euro symbol
by combining a “C” with a “=”.

```

926 \def\tc@fake@euro#1{%
927   \leavevmode
928   \PackageInfo{textcomp}{Faking \noexpand#1 for font family
929                           \f@family\MessageBreak in TS1 encoding}%
930   \valign{##\cr
931     \vfil\hbox to 0.07em{\dimen@\f@size\p@
932                               \math@fontsfalse
933                               \fontsize{.7\dimen@}\z@\selectfont=\hss}%
934     \vfil\cr%
935     \hbox{C}\crcr
936   }%
937 }

```

(End of 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 of 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}
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^AT_EX 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 ‘o’ 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 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}{pp1}      {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}{ugg}      {5}
1170 \DeclareEncodingSubset{TS1}{u18}      {4}
1171 \DeclareEncodingSubset{TS1}{u19}      {4} % LuxiSans, one day)
1172 \DeclareEncodingSubset{TS1}{augie}    {5}
1173 \DeclareEncodingSubset{TS1}{dayrom}   {3}
1174 \DeclareEncodingSubset{TS1}{dayroms} {3}
1175 \DeclareEncodingSubset{TS1}{pxr}      {0}
1176 \DeclareEncodingSubset{TS1}{pxss}     {0}
1177 \DeclareEncodingSubset{TS1}{pxtt}     {0}
1178 \DeclareEncodingSubset{TS1}{txr}      {0}
1179 \DeclareEncodingSubset{TS1}{txss}     {0}
1180 \DeclareEncodingSubset{TS1}{txtt}     {0}

```

Latin Modern and TeX Gyre:

```
1181 \DeclareEncodingSubset{TS1}{lmr}      {0}
1182 \DeclareEncodingSubset{TS1}{lmdh}     {0}
1183 \DeclareEncodingSubset{TS1}{lmss}      {0}
1184 \DeclareEncodingSubset{TS1}{lmssq}     {0}
1185 \DeclareEncodingSubset{TS1}{lmvtt}     {0}
1186 \DeclareEncodingSubset{TS1}{lmtt}      {0}
1187 \DeclareEncodingSubset{TS1}{qhv}       {0}
1188 \DeclareEncodingSubset{TS1}{qag}       {0}
1189 \DeclareEncodingSubset{TS1}{qbk}       {0}
1190 \DeclareEncodingSubset{TS1}{qcr}       {0}
1191 \DeclareEncodingSubset{TS1}{qcs}       {0}
1192 \DeclareEncodingSubset{TS1}{qpl}       {0}
1193 \DeclareEncodingSubset{TS1}{qtm}       {0}
1194 \DeclareEncodingSubset{TS1}{qzc}       {0}
1195 \DeclareEncodingSubset{TS1}{qhvc}     {0}
```

Fourier-GUTenberg:

```
1196 \DeclareEncodingSubset{TS1}{futs}     {4}
1197 \DeclareEncodingSubset{TS1}{futx}     {4}
1198 \DeclareEncodingSubset{TS1}{futj}     {4}
```

Y&Y's Lucida Bright

```
1199 \DeclareEncodingSubset{TS1}{hlh}      {3}
1200 \DeclareEncodingSubset{TS1}{hls}      {3}
1201 \DeclareEncodingSubset{TS1}{hlst}     {3}
```

The remaining settings for Lucida are conservative: the following fonts contain the `\textohm` character but not the `\texteuro`, i.e., belong to neither subset 4 nor subset 3. If you want to use the `\textohm` with these fonts copy these definition to `textcomp.cfg` and change the subset to 3. However in that case make sure that you do not use the `\texteuro`.

```
1202 \DeclareEncodingSubset{TS1}{hlct}     {5}
1203 \DeclareEncodingSubset{TS1}{hlx}      {5}
1204 \DeclareEncodingSubset{TS1}{hlce}     {5}
1205 \DeclareEncodingSubset{TS1}{hlcn}     {5}
1206 \DeclareEncodingSubset{TS1}{hlcw}     {5}
1207 \DeclareEncodingSubset{TS1}{hlcf}     {5}
```

Other commercial families...

```
1208 \DeclareEncodingSubset{TS1}{pplx}     {3}
1209 \DeclareEncodingSubset{TS1}{pplj}     {3}
1210 \DeclareEncodingSubset{TS1}{ptmx}     {4}
1211 \DeclareEncodingSubset{TS1}{ptmj}     {4}
```

If the file `textcomp.cfg` exists it will be loaded at this point. This allows to define further subset encodings for font families not covered by default.

```
1212 \InputIfFileExists{textcomp.cfg}
1213   {\PackageInfo{textcomp}{Local configuration file used}}{}
1214 \fi
1215 </TS1oldsty>
```

File F

ltpageno.dtx

1 Page Numbering

Page numbers are produced by a page counter, used just like any other counter. The only difference is that `\c@page` contains the number of the next page to be output (the one currently being produced), rather than one minus it. Thus, it is normally initialized to 1 rather than 0. `\c@page` is defined to be `\count0`, rather than a count assigned by `\newcount`.

`\pagenumbering` The user sets the page number style with the `\pagenumbering{<foo>}` command, which sets the page counter to 1 and defines `\thepage` to be `\foo`. For example, `\pagenumbering{roman}` causes pages to be numbered i, ii, etc.

```
1  {*2ekernel}
2  \message{page nos.,}
3  \countdef\c@page=0 \c@page=1
4  \def\cl@page{}
5  \def\pagenumbering#1{%
6    \global\c@page \cne \gdef\thepage{\csname \#1\endcsname
7    \c@page}}
8  {/2ekernel}
```

File G

ltxref.dtx

1 Cross Referencing

The user writes `\label{<foo>}` to define the following cross-references:

`\ref*<{<foo>}`: value of most recently incremented referenceable counter. in the current environment. (Chapter, section, theorem, footnote and enumeration counters and other counters stepped with `\refstepcounter` are referenceable.)

`\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.

Note: the starred versions `\ref*` and `\pageref*` are provided to align with the use of `hyperref`. Without `hyperref` (or some other package using the starred form) the star is simply ignored.

Note: starting with 2023-06-01 `\label` stores also the current value of `\@currentlabelname` which should typically contain a (sanitized) title. (A reference command `\nameref` is provided by the `nameref` package.) `\label` also stores `\@currentHref` which if set should refer to a target name for links. This value is set and used by `hyperref`. Unlike the other values `\@currentHref` should be set globally. A fifth value `\@kernel@reserved@label@data` is reserved for the kernel to allow future extensions of the cross-reference system.

1.1 Cross Referencing

Historical L^AT_EX 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)}%
```

```
{eval(\@currentlabelname)}{eval(\@currentHref)}{eval(\@kernel@reserved@label@data)}}

\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^AT_EX 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 This does not save on name-space (since \G@refundefinedfalse was never needed) but \G@refundefined true it does make the implementation of such one-way switches more consistent. The extra macro to make the change is used since this change appears several times.

Note despite its name, \G@refundefinedtrue does *not* correspond to an \if command, and there is no matching ... false. It would be more natural to call the command \G@refundefined (as inspection of the change log will reveal) but unfortunately such a change would break any package that had defined a \ref-like command that mimicked the definition of \ref, calling \G@refundefinedtrue. Inspection of the TeX 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 of definition for `\G@refundefinedtrue` and `\@refundefined`.)

<code>\ref</code>	Referencing a <code>\label</code> . RmS 91/10/25: added a few extra <code>\reset@font</code> , as suggested by Bernd Raichle
<code>\pageref</code>	RmS 92/08/14: made <code>\ref</code> and <code>\pageref</code> robust RmS 93/09/08: Added setting of <code>refundefined</code> switch.
<code>\@setref</code>	<pre> 10 </2ekernel> 11 <*2ekernel latexrelease> 12 <latexrelease>\IncludeInRelease{2023/06/01}% 13 <latexrelease> {\@kernel@sref}{store five arguments}% 14 \def\@setref#1#2#3{% 15 \ifx#1\relax 16 \protect\G@refundefinedtrue 17 \nfss@text{\reset@font\bfseries ??}% 18 \@latex@warning{Reference '#3' on page \thepage \space 19 undefined}% 20 \else 21 \expandafter#2#1\empty\empty\empty\null 22 \fi} 23 \long\def\@firstoffive#1#2#3#4#5{#1} 24 \long\def\@secondoffive#1#2#3#4#5{#2} 25 \def\@kernel@sref#1{\expandafter\@setref\csname r@#1\endcsname\@firstoffive{#1}} 26 \def\@kernel@spageref#1{\expandafter\@setref\csname r@#1\endcsname 27 \@secondoffive{#1}} 28 <latexrelease>\EndIncludeInRelease 29 <latexrelease>\IncludeInRelease{2022/06/01}% 30 <latexrelease> {\@kernel@sref}{store five arguments}% 31 <latexrelease>\def\@setref#1#2#3{% 32 <latexrelease> \ifx#1\relax 33 <latexrelease> \protect\G@refundefinedtrue 34 <latexrelease> \nfss@text{\reset@font\bfseries ??}% 35 <latexrelease> \@latex@warning{Reference '#3' on page \thepage \space 36 undefined}% 37 <latexrelease> \else 38 <latexrelease> \expandafter#2#1\null 39 <latexrelease> \fi} 40 <latexrelease>\let\@firstoffive\undefined 41 <latexrelease>\let\@secondoffive\undefined 42 <latexrelease>\def\@kernel@sref#1{\expandafter\@setref\csname r@#1\endcsname\@firstoftwo{#1}} 43 <latexrelease>\def\@kernel@spageref#1{\expandafter\@setref\csname r@#1\endcsname 44 \@secondoftwo{#1}} 45 <latexrelease>\EndIncludeInRelease 46 <latexrelease>\IncludeInRelease{0000/00/00}% </pre>

```

47  \begin{macro}{\@kernel@sref}{store five arguments}%
48  \def\@setref#1#2#3{%
49    \ifx#1\relax
50    \protect\G@refundefinedtrue
51    \nfss@text{\reset@font\bfseries ??}%
52    \@latex@warning[Reference '#3' on page \thepage \space
53    undefined]%
54  \else
55    \expandafter#2#1\null
56  \fi}
57  \let\@firstoffive\undefined
58  \let\@secondoffive\undefined
59  \let\@kernel@sref\undefined
60  \let\@kernel@spageref\undefined
61  \EndIncludeInRelease
62  \IncludeInRelease{2022/06/01}%
63  \begin{macro}{\@ref}{Add starred reference commands}%
64  \let\@kernel@ref\@kernel@sref
65  \let\@kernel@pageref\@kernel@spageref
66  \NewDocumentCommand{\ref}{s}
67    {\IfBooleanTF{#1}{\@kernel@sref}{\@kernel@ref}}
68  \NewDocumentCommand{\pageref}{s}
69    {\IfBooleanTF{#1}{\@kernel@spageref}{\@kernel@pageref}}%

```

As the commands are now protected we also need expandable versions for use in `\ifthenelse`:

```

70  \def\@kernel@pageref@exp#1{\csname cs_if_exist:cTF\endcsname
71    {r@#1}\{\csname tl_item:cn\endcsname{r@#1}{2}\}{0}\}
72  \def\@kernel@ref@exp#1{\csname cs_if_exist:cTF\endcsname
73    {r@#1}\{\csname tl_item:cn\endcsname{r@#1}{1}\}{0}\}
74  (/2ekernel | \begin{macro}{\@kernel}{\@release})
75  \EndIncludeInRelease
76  \IncludeInRelease{0000/00/00}%
77  \begin{macro}{\@ref}{Add starred reference commands}%
78  \def\@ref#1{\expandafter\@setref\csname r@#1\endcsname\@firstoftwo{#1}}
79  \def\@pageref#1{\expandafter\@setref\csname r@#1\endcsname
80    \secondoftwo{#1}}%
81  \begin{macro}{\@pageref}{\@ref}
82  \EndIncludeInRelease
83  (*2ekernel)

```

(End of 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 `\bibcrite`. 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.

```

84  \def\@newl@bel#1#2#3{%
85    \@ifundefined{#1#2}%
86      \relax
87      \gdef\@multiplelabels{%
88        \@latex@warning@no@line{There were multiply-defined labels}}%

```

```

89      \@latex@warning@no@line{Label '#2' multiply defined}%
90  \global\@namedef{\@newl@bel}{#3}%
91  \def\newlabel{\@newl@bel r}%
92  \onlypreamble\@newl@bel

```

(End of 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.

```

93  \let \@multiplelabels \relax

```

(End of 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}}.\romannumeral{section}}
```

will cause a \label{bar} command to define \ref{bar} to expand to something like \foo{4.d}. Change made 20 Jul 88.

```

94  </2ekernel>
95  {*2ekernel | latexrelease}
96  <latexrelease>\IncludeInRelease{2023/06/01}%
97  <latexrelease>          {\label}{store five label arguments}%
98  \providecommand{\currentlabelname}{}
99  \providecommand{\currentHref}{}
100 \providecommand{\kernel@reserved@label@data}{}
101 \NewHookWithArguments{label}{1}
102 \def\label#1{\bsphack
103   \begingroup
104   \UseHookWithArguments{label}{1}{#1}%
105   \protected@write{\auxout}{}
106   {\string\newlabel{#1}{\currentlabelname{\thepage}}%
107    {\currentlabelname{\currentHref}{\kernel@reserved@label@data}}}%
108   \endgroup
109   \esphack
110 <latexrelease>\EndIncludeInRelease

```

(End of definition for \label.)

```

111 <latexrelease>\IncludeInRelease{2022/06/01}%
112 <latexrelease>          {\Ref}{Add starred version}%

```

\refstepcounter Step the counter and allow for labels to point to its current value.

```

113 \def\@currentcounter{}%
114 \def\refstepcounter#1{\stepcounter{#1}%
115   \edef\@currentcounter{#1}%
116   \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.

```
117     {\csname p@\#1\expandafter\endcsname\csname the#1\endcsname}%
118 }
```

(End of 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.

```
119 \def\labelformat#1{\expandafter\def\csname p@\#1\endcsname##1}
```

(End of 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`.³⁵

```
120 \def\@kernel@Ref#1{\protected@edef\@tempa{\@kernel@ref{#1}}%
121     \expandafter\MakeUppercase\@tempa}
122 \def\@kernel@sRef#1{\protected@edef\@tempa{\@kernel@sref{#1}}%
123     \expandafter\MakeUppercase\@tempa}
124 \NewDocumentCommand\Ref{s}
125   {\IfBooleanTF{#1}{\@kernel@sRef}{\@kernel@Ref}}
```

(End of definition for `\Ref`.)

```
126 </2ekernel | latexrelease>
127 <latexrelease>\EndIncludeInRelease
128 <latexrelease>\IncludeInRelease{0000/00/00}%
129 <latexrelease>          {\label}{store five label arguments}%
130 <latexrelease>\let\@currenttitle\@undefined
131 <latexrelease>\let\@currenttarget\@undefined
132 <latexrelease>\let\@kernel@currentdata\@undefined
133 <latexrelease>\def\label#1{\@bsphack
134 <latexrelease>  \protected@write\@auxout{}{%
135 <latexrelease>    {\string\newlabel{#1}{{\@currentlabel}{\thepage}}}}%
136 <latexrelease>  \@esphack}
137 <latexrelease>\EndIncludeInRelease
138 <latexrelease>\IncludeInRelease{2020/10/01}%
139 <latexrelease>          {\Ref}{Add starred version}%
140 <latexrelease>\def\@currentcounter{}
141 <latexrelease>\def\refstepcounter#1{\stepcounter{#1}}%
142 <latexrelease>  \edef\@currentcounter{#1}%
143 <latexrelease>  \protected@edef\@currentlabel
144 <latexrelease>    {\csname p@\#1\expandafter\endcsname\csname the#1\endcsname}%
145 <latexrelease>}
```

```
146 <latexrelease>\def\labelformat#1{\expandafter\def\csname p@\#1\endcsname##1}
147 <latexrelease>\DeclareRobustCommand\Ref[1]{\protected@edef\@tempa{\ref{#1}}}%
```

³⁵There is one problem with this approach: the braces are kept in a normal `\ref` which might spoil kerning. Perhaps one day this needs redoing.

```

148 〈latexrelease〉 \expandafter\MakeUppercase\@tempa}
149 〈latexrelease〉\EndIncludeInRelease
150 〈latexrelease〉\IncludeInRelease{2019/10/01}%
151 〈latexrelease〉 \refstepcounter{Add \labelformat and \Ref}%
152 〈latexrelease〉\let\@currentcounter\@undefined
153 〈latexrelease〉\def\refstepcounter#1{\stepcounter{#1}%
154 〈latexrelease〉 \protected@edef\@currentlabel
155 〈latexrelease〉 {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}%
156 〈latexrelease〉}
157 〈latexrelease〉\def\labelformat#1{\expandafter\def\csname p@#1\endcsname##1}
158 〈latexrelease〉\DeclareRobustCommand\Ref[1]{\protected@edef\@tempa{\ref{#1}}}%
159 〈latexrelease〉 \expandafter\MakeUppercase\@tempa}
160 〈latexrelease〉\EndIncludeInRelease
161 〈latexrelease〉\IncludeInRelease{0000/00/00}%
162 〈latexrelease〉 \refstepcounter{Add \labelformat and \Ref}%
163 〈latexrelease〉
164 〈latexrelease〉\def\refstepcounter#1{\stepcounter{#1}%
165 〈latexrelease〉 \protected@edef\@currentlabel
166 〈latexrelease〉 {\csname p@#1\endcsname\csname the#1\endcsname}%
167 〈latexrelease〉}
168 〈latexrelease〉\let\labelformat\@undefined
169 〈latexrelease〉\let\Ref\@undefined
170 〈latexrelease〉
171 〈latexrelease〉\EndIncludeInRelease
172 〈*2ekernel〉

```

`\@currentlabel` Default for `\label` commands that come before any environment.

```
173 \def\@currentlabel{}
```

(End of definition for `\@currentlabel`.)

```
174 〈/2ekernel〉
```

File H

ltmiscen.dtx

1 Miscellaneous Environments

This section implements the basic environment mechanism, and also a few specific environments including `document`, The math environments and related commands, the ‘flushing’ environments, (`center`, `flushleft`, `flushright`), and `verbatim`.

```
1  <*2ekernel>
2  \message{environments,}
```

1.1 Environments

`\begin{foo}` and `\end{foo}` are used to delimit environment `foo`.

`\begin{foo}` starts a group and calls `\foo` if it is defined, otherwise it does nothing.

`\end{foo}` checks to see that it matches the corresponding `\begin` and if so, it calls `\endfoo` and does an `\endgroup`. Otherwise, `\end{foo}` does nothing.

If `\end{foo}` needs to ignore blanks after it, then `\endfoo` should globally set the `@ignore` switch true with `\@ignoretrue` (this will automatically be global).

NOTE: `\@end` is defined to be the `\end` command of TeX82.

`\enddocument` is the user’s command for ending the manuscript file.

`\stop` is a panic button — to end TeX in the middle.

Historical L^AT_EX 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^AT_EX 2.09 comments.

\@currenvir The name of the current environment. Initialized to document to so that \end{document} works correctly.

```

3 \def\@currenvir{document}
```

(End of 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 of definition for \if@ignore, \@ignoretrue, and \@ignorefalse.)

\ignorespacesafterend

```

7 \let\ignorespacesafterend\@ignoretrue
```

(End of definition for \ignorespacesafterend.)

\end{document})

```

8 </2ekernel>
9 {*2ekernel | latexrelease}
10 <latexrelease>\IncludeInRelease{2020/10/01}%
11 <latexrelease>           {\enddocument}{Use Hooks}%
12 \def\enddocument{%
```

The \end{document} hook is executed first. If necessary it can contain a \clearpage to output dangling floats first. In this position it can also contain something like \end{foo} so that the whole document effectively starts and ends with some special environment. However, this must be used with care, eg if two applications would use this without knowledge of each other the order of the environments will be wrong after all. \AtEndDocument is redefined at this point so that and such commands that get into the hook do not chase their tail...

```

13 \@kernel@before@enddocument
14 \UseOneTimeHook{enddocument}%
15 \@kernel@after@enddocument
```

```

16  \@checkend{document}%
17  \clearpage
18  \UseOneTimeHook{enddocument/afterlastpage}%
19  \@kernel@after@enddocument@afterlastpage
20  \begingroup
21  \if@filesw
22    \immediate\closeout\mainaux
23    \let\@setckpt\@gobbletwo
24    \let\@newl@bel\@testdef

```

The previous line is equiv to setting

```

\def\newlabel{\@testdef r}%
\def\bibcite{\@testdef b}%

```

We use `\@@input` to load the .aux file, so that it doesn't show up in the list of files produced by `\listfiles`.

```

25  \tempswafalse
26  \makeatletter \@@input\jobname.aux
27  \fi
28  \UseOneTimeHook{enddocument/afteraux}%

```

Next hook is expect to contain only code for writing info messages on the terminal.

```

29  \UseOneTimeHook{enddocument/info}%
30  \endgroup
31  \UseOneTimeHook{enddocument/end}%
32  \deadcycles{z@\@@end}

```

The public hooks used in `\enddocument`:

```

33 \NewHook{enddocument}
34 \NewHook{enddocument/afterlastpage}
35 \NewHook{enddocument/afteraux}
36 \NewHook{enddocument/info}
37 \NewHook{enddocument/end}

```

This is one of the few places where we already add data and rules to a hook already in the kernel.

```

38 \AddToHook{enddocument/info}[kernel/filelist]{\@dofilelist}
39 \AddToHook{enddocument/info}[kernel/warnings]{\@enddocument@kernel@warnings}%
40 \AddToHook{enddocument/info}[kernel/release]{%
41   \let\show@release@info\wlog
42   \show@release@info{ ****}%
43   \the\LaTeXReleaseInfo
44   \show@release@info{ ****}}%
45
46 \DeclareHookRule{enddocument/info}{kernel/release}{before}{kernel/filelist}
47 \DeclareHookRule{enddocument/info}{kernel/filelist}{before}{kernel/warnings}

```

(End of definition for `\enddocument`.)

`\@enddocument@kernel@warnings`

```

48 \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.

```

49 \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.

```
50     \@font@warning{Size substitutions with differences\MessageBreak
51         up to \font@submax\space have occurred.\@gobbletwo}%
52     \fi
```

The macro `\@defaultsubs` is initially `\relax` but gets redefined to produce a warning if there have been some default font substitutions.

```
53     \@defaultsubs
```

The macro `\@refundefined` is initially `\relax` but gets redefined to produce a warning if there are undefined refs.

```
54     \@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 `varioref` that generates labels on the fly), we suppress this message.

```
55     \if@filesw
56         \ifx \@multiplelabels \relax
57             \if@tempswa
58                 \@latex@warning{no@line{Label(s) may have changed.
59                             Rerun to get cross-references right}%
60             \fi
61         \else
62             \@multiplelabels
63         \fi
64         \ifx \@extra@page@added \relax
65             \@latex@warning{no@line{Temporary extra page added at the end.
66                             Rerun to get it removed}%
67         \fi
68     \fi
69 }
```

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.

```
68     \fi
69 }

(End of definition for \enddocument@kernel@warnings.)

70 </2ekernel | latexrelease>
71 <latexrelease>\EndIncludeInRelease
72 <latexrelease>\IncludeInRelease{0000/00/00}%
73 <latexrelease>          {\enddocument}{Use Hooks}%
74 <latexrelease>
75 <latexrelease>\def\enddocument{%
76 <latexrelease>    \let\AtEndDocument\@firstofone
77 <latexrelease>    \enddocumenthook
78 <latexrelease>    \checkend{document}%
79 <latexrelease>    \clearpage
80 <latexrelease>    \begin{group}
81 <latexrelease>        \if@filesw
82 <latexrelease>            \immediate\closeout\mainaux
83 <latexrelease>            \let\@setckpt\@gobbletwo
84 <latexrelease>            \let\@newl@bel\@testdef
85 <latexrelease>            \tempswafalse
86 <latexrelease>            \makeatletter \@@input\jobname.aux
```

```

87  ⟨latexrelease⟩      \fi
88  ⟨latexrelease⟩      \@dofilelist
89  ⟨latexrelease⟩      \ifdim \font@submax > \fontsubfuzz \relax
90  ⟨latexrelease⟩          \@font@warning{Size substitutions with differences\MessageBreak
91  ⟨latexrelease⟩              up to \font@submax\space have occurred.\@gobbletwo}%
92  ⟨latexrelease⟩      \fi
93  ⟨latexrelease⟩      \@defaultsubs
94  ⟨latexrelease⟩      \@refundefined
95  ⟨latexrelease⟩      \if@filesw
96  ⟨latexrelease⟩          \ifx \multiplelabels \relax
97  ⟨latexrelease⟩              \if@tempswa
98  ⟨latexrelease⟩                  \@latex@warning@no@line{Label(s) may have changed.
99  ⟨latexrelease⟩                      Rerun to get cross-references right}%
100 ⟨latexrelease⟩      \fi
101 ⟨latexrelease⟩      \else
102 ⟨latexrelease⟩          \multiplelabels
103 ⟨latexrelease⟩      \fi
104 ⟨latexrelease⟩      \fi
105 ⟨latexrelease⟩      \endgroup
106 ⟨latexrelease⟩      \deadcycles{z@\@end}
107 ⟨latexrelease⟩
108 ⟨latexrelease⟩\let\enddocument@kernel@warnings\undefined
109 ⟨latexrelease⟩
110 ⟨latexrelease⟩\EndIncludeInRelease
111 ⟨*2ekernel⟩

```

\@kernel@before@enddocument The \@kernel@before@enddocument hook is slightly different because we initialize it with \par so that \enddocument always returns to vertical mode as its first action.

```

112 ⟨/2ekernel⟩
113 ⟨*2ekernel | latexrelease⟩
114 ⟨latexrelease⟩\IncludeInRelease{2021/06/01}%
115 ⟨latexrelease⟩          {\@kernel@before@enddocument}{kernel before hook}%
116 \def\@kernel@before@enddocument{\par}
117 ⟨/2ekernel | latexrelease⟩
118 ⟨latexrelease⟩\EndIncludeInRelease

```

The rollback code renders it harmless.

```

119 ⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
120 ⟨latexrelease⟩          {\@kernel@before@enddocument}{kernel before hook}%
121 ⟨latexrelease⟩
122 ⟨latexrelease⟩\let\@kernel@before@enddocument\empty
123 ⟨latexrelease⟩
124 ⟨latexrelease⟩\EndIncludeInRelease
125 ⟨*2ekernel⟩

```

(*End of definition for \@kernel@before@enddocument.*)

\@testdef

```

126 \def\@testdef #1#2#3{%
127   \def\reserved@a{#3}\expandafter \ifx \csname #1#2\endcsname
128   \reserved@a \else \@tempswatru\fi}

```

(*End of 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.

```
129  </2ekernel>
130  <*2ekernel | latexrelease>
131  <| latexrelease>\IncludeInRelease{2018/12/01}%
132  <| latexrelease>          {\protected@file@percent}{Mask line endings}%
133  \protected\def\protected@file@percent{}%
```

(End of 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.

```
134  \catcode`^\^A=9
135  \long\gdef\add@percent@to@temptokena
136    #1\protected@file@percent#2\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.

```
137  {\expandafter\ifx\expandafter X\detokenize{#2}X\expandafter\dont@add@percent@to@temptokena
138    \expandafter\do@add@percent@to@temptokena\fi{#1}}
139  \long\def\dont@add@percent@to@temptokena#1{%
140    \@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 :-)

```

141 \begingroup
142 \catcode`\%=12
143 \catcode`\^^A=14
144 \long\gdef\do@add@percent@to@temptokena#1{\@temptokena\expandafter{\#1%``A

```

Can't be on the same line as the `%` — see above.

```

145   }
146 \endgroup

```

(*End of definition for `\add@percent@to@temptokena`.*)

`\@writefile`

```

147 \long\def\@writefile#1#2{%
148   \@ifundefined{tf@#1}\relax
149   {%

```

If we write to the file we first prepare #2 using `\add@percent@to@temptokena` and then write the token register out.

```

150   \add@percent@to@temptokena
151     \empty#2\protected@file@percent
152     \add@percent@to@temptokena
153     \immediate\write\csname tf@#1\endcsname{\the\@temptokena}%
154   }%
155 }

156 (//2ekernel | latexrelease)
157 <latexrelease>\EndIncludeInRelease
158 <latexrelease>\IncludeInRelease{0000/00/00}%
159 <latexrelease>          {\protected@file@percent}{Mask line endings}%
160 <latexrelease>\let\protected@file@percent\@undefined
161 <latexrelease>\let\add@percent@to@temptokena\@undefined
162 <latexrelease>\let\do@add@percent@to@temptokena\@undefined
163 <latexrelease>\let\dont@add@percent@to@temptokena\@undefined
164 <latexrelease>\long\def\@writefile#1#2{%
165   \@ifundefined{tf@#1}\relax
166   {%
167     \empty{#2}%
168     \immediate\write\csname tf@#1\endcsname{\the\@temptokena}%
169   }%
170 <latexrelease>\EndIncludeInRelease
171 <2ekernel>

```

(*End of definition for `\@writefile`.*)

`\stop`

```

172 \def\stop{\clearpage\deadcycles\z@\let\par\@@par\@@end}

```

(End of definition for \stop.)

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

173 \everypar{\@nодокумент} %% To get an error if text appears before the
174 \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^AT_EX 2.09 comments.

```

\begin{  

175  </2ekernel>  

176  {*2ekernel | latexrelease}  

177  {latexrelease}\IncludeInRelease{2020/10/01}%  

178  {latexrelease}           {\begin}{Use hook system}%  

179  \DeclareRobustCommand*\begin[1]{%  

180    \UseHook{env/#1/before}}%  

181  \@ifundefined{#1}{%  

182    {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%  

183    {\def\reserved@a{\def\currenvir{#1}}%  

184      \edef\@currenvline{\on@line}}%  

185      \execute@begin@hook{#1}}%  

186      \csname #1\endcsname}}%  

187  \@ignorefalse  

188  \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.

```

189 \def\execute@begin@hook #1{%
190   \expandafter\ifx\csname #1\endcsname\document
191   \endgroup
192   \gdef\execute@begin@hook##1{\UseHook{env/##1/begin}}%
193   \expl@@initialize@all@@
194 }

```

If this is an environment before `\begin{document}` we just run the hook so this can be outside the test.

```

195 \UseHook{env/#1/begin}%
196 }

```

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).

```

197 \edef\end
198   {\unexpanded{%
199     \romannumeral
200       \ifx\protect\@typeset@protect
201       \expandafter      %1
202       \expandafter      %2
203       \expandafter      %1
204       \expandafter      %3 expands the \csname inside \end<space>
205       \expandafter      %1
206       \expandafter      %2 expands \end<space>
207       \expandafter      %1      expands the \else
208       \z@
209     \else
210       \expandafter\z@\expandafter\protect
211     \fi
212   }%
213   \expandafter\noexpand\csname end \endcsname
214 }

```

Version that adds hooks (so different from the 2019 version). It fixes tlb3722 but the change should perhaps be made in `tabularx` instead.

```

215  \Onamedef{end }#1{%
216    \romannumeral
217    \IfHookEmptyTF{env/#1/end}%
218    {\expandafter\z@}%
219    {\z@\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  }

```

Version without the fix for tlb3722 for the record:

```

225  \% \Onamedef{end }#1{%
226  %   \UseHook{env/#1/end}}%
227  %   \csname end#1\endcsname\@checkend{#1}%
228  %   \expandafter\endgroup\if@endpe\@doendpe\fi
229  %   \UseHook{env/#1/after}%
230  %   \if@ignore\@ignorefalse\ignorespaces\fi}%
231  </2ekernel | latexrelease>
232  <| latexrelease>\EndIncludeInRelease
233  <| latexrelease>\IncludeInRelease{2019/10/01}%
234  <| latexrelease>           {\begin}{\end robust}%
235  <| latexrelease>\DeclareRobustCommand\begin[1]{%
236  <| latexrelease> \@ifundefined{#1}%
237  <| latexrelease>   {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
238  <| latexrelease>   {\def\reserved@a{\def\currenvir{#1}}%
239  <| latexrelease>     \edef\currenvline{\on@line}%
240  <| latexrelease>     \csname #1\endcsname}}%
241  <| latexrelease> \@ignorefalse
242  <| 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.

```

243  \% \edef\begin
244  %   {\unexpanded{%
245  %     \ifx\protect\@typeset@protect
246  %       \expandafter\@gobble
247  %     \fi
248  %     \protect
249  %   }%
250  %   \expandafter\noexpand\csname begin \endcsname
251  % }
252  \% \Onamedef{begin }#1{%
253  %   \ifundefined{#1}%
254  %     {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
255  %     {\def\reserved@a{\def\currenvir{#1}}%
256  %       \edef\currenvline{\on@line}%
257  %       \csname #1\endcsname}}%
258  %   \@ignorefalse
259  %   \begingroup\@endpefalse\reserved@a}

```

`\end` While `\begin` was made robust simply by using `\DeclareRobustCommand` we need to be a bit more subtle with `\end` as there are packages out there that try to look into the top-level contents of `\end{foo}` (that is at the expansion of `\endfoo`) to see if it contains certain macros. This is done by hitting `\end{foo}` with three `\expandafters`, the first to get

```
\csname endfoo\endcsname      @checkend{foo}%
etc.
```

the second to expand the `\csname`, i.e., to get to

```
\endfoo      @checkend{foo}%
etc.
```

and the third to finally get to the top-level content of `\endfoo`, i.e.

```
<top-level content of \endfoo> @checkend{foo}%
etc.
```

Therefore a robust replacement should produce the same results after three expansions (there first is obviously different).

Basically the definition of `\end` should either produce `\protect\end_` (when not doing typesetting) or it should produce `\end_` (without the `\protect`) when doing typesetting. Furthermore, it should (when in typesetting mode) show exactly the same result as `\end_` (which is the original fragile definition of `\end`) when you expand either of them twice, i.e.,

```
\endfoo      @checkend{foo}%
etc.
```

That is achieved with the code below (which is worth studying carefully).

There is some trickery involved here: in particular we use `\romannumeral` to change a single expansion into three successive expansions in one go. That primitive expands until it has scanned a number (0 in this case, so it doesn't produce any output) and so it allows us to place arbitrary many `\expandafters` inside that are all going to be executed when `\romannumeral` is hit by a single `\expandafter`.

```

260 <|latexrelease>\edef\end
261 <|latexrelease>  {\unexpanded{%
262 <|latexrelease>    \romannumeral
263 <|latexrelease>      \ifx\protect\@typeset@protect
264 <|latexrelease>        \expandafter      %1
265 <|latexrelease>        \expandafter      %2
266 <|latexrelease>        \expandafter      %1
267 <|latexrelease>        \expandafter      %3 expands the \csname inside \end<space>
268 <|latexrelease>        \expandafter      %1
269 <|latexrelease>        \expandafter      %2 expands \end<space>
270 <|latexrelease>        \expandafter      %1      expands the \else
271 <|latexrelease>          \z@
272 <|latexrelease>        \else
273 <|latexrelease>          \expandafter\z@\expandafter\protect
274 <|latexrelease>        \fi
275 <|latexrelease>  }%
276 <|latexrelease>  \expandafter\noexpand\csname end \endcsname
277 <|latexrelease> }
```

And here is the original definition of `\end` the way it was in L^AT_EX for several decades now hidden in `\end_`.

```
278 <|latexrelease>\@namedef{end }#1%
```

```

279 〈latexrelease〉 \csname end#1\endcsname\@checkend{#1}%
280 〈latexrelease〉 \expandafter\endgroup\if@endpe\@doendpe\fi
281 〈latexrelease〉 \if@ignore\@ignorefalse\ignorespaces\fi}
282 〈latexrelease〉\EndIncludeInRelease

```

An here the rollback in case that is ever needed.

```

283 〈latexrelease〉\IncludeInRelease{0000/00/00}%
284 〈latexrelease〉 \begin{Making \begin/\end robust}%
285 〈latexrelease〉\def\begin#1{%
286 〈latexrelease〉 \@ifundefined{#1}{%
287 〈latexrelease〉 \def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}{%
288 〈latexrelease〉 \def\reserved@a{\def\currenvir{#1}}%
289 〈latexrelease〉 \edef\currenvline{\on@line}%
290 〈latexrelease〉 \csname #1\endcsname}%
291 〈latexrelease〉 \@ignorefalse
292 〈latexrelease〉 \begingroup\@endpefalse\reserved@a
293 〈latexrelease〉\def\end#1{%
294 〈latexrelease〉 \csname end#1\endcsname\@checkend{#1}%
295 〈latexrelease〉 \expandafter\endgroup\if@endpe\@doendpe\fi
296 〈latexrelease〉 \if@ignore\@ignorefalse\ignorespaces\fi}
297 〈latexrelease〉

```

Also undo the internal commands as some packages unfortunately test for their existence instead of using `\IfFormatAtLeastTF`.

```

298 〈latexrelease〉\expandafter\let\csname begin \endcsname\@undefined
299 〈latexrelease〉\expandafter\let\csname end \endcsname\@undefined
300 〈latexrelease〉
301 〈latexrelease〉\EndIncludeInRelease
302 (*2ekernel)

```

(End of definition for `\begin` and `\end`.)

`\@checkend`

```

303 \def\@checkend#1{\def\reserved@a{#1}\ifx
304 \reserved@a\currenvir \else\badend{#1}\fi}

```

(End of 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.

```

305 \let\@currenvline\@empty

```

(End of definition for `\@currenvline`.)

`\AtBeginEnvironment` We provide 4 high-level hook interfaces directly, the others only when etoolbox is loaded
`\AtEndEnvironment`
`\BeforeBeginEnvironment`
`\AfterEndEnvironment`

```

306 (/2ekernel)
307 (*2ekernel | latexrelease)
308 〈latexrelease〉\IncludeInRelease{2020/10/01}%
309 〈latexrelease〉 \AtBeginEnvironment{Hooks for environments}%
310 \newcommand\AtBeginEnvironment[2][]{\AddToHook{env/#2/begin}[]{#1}}
311 \newcommand\AtEndEnvironment[2][]{\AddToHook{env/#2/end}[]{#1}}
312 \newcommand\BeforeBeginEnvironment[2][]{\AddToHook{env/#2/before}[]{#1}}
313 \newcommand\AfterEndEnvironment[2][]{\AddToHook{env/#2/after}[]{#1}}

```

```

314  </2ekernel | latexrelease>
315  <latexrelease>\EndIncludeInRelease
316  <latexrelease>\IncludeInRelease{0000/00/00}%
317  <latexrelease>                      {\AtBeginEnvironment}{Hooks for environments}%
318  <latexrelease>
319  <latexrelease>\let\AtBeginEnvironment\@undefined
320  <latexrelease>\let\AtEndEnvironment\@undefined
321  <latexrelease>\let\BeforeBeginEnvironment\@undefined
322  <latexrelease>\let\AfterEndEnvironment\@undefined
323  <latexrelease>
324  <latexrelease>\EndIncludeInRelease
325  <*2ekernel>

```

(End of definition for `\AtBeginEnvironment` and others. These functions are documented on page 214.)

1.2 Center, Flushright, Flushleft

```
326 \message{center,}
```

Historical L^AT_EX 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^AT_EX 2.09 comments.

```

\@centercr
327 </2ekernel>
328 <*2ekernel | latexrelease>
329 <latexrelease>\IncludeInRelease{2020/02/02}%
330 <latexrelease>                      {\@centercr}{Make robust}%
331 \protected\def\@centercr{\ifhmode \unskip\else \nolnerr\fi
332           \par\@ifstar{\nobreak\@xcentercr}\@xcentercr}
333 </2ekernel | latexrelease>

```

```

334 〈\latexrelease〉\EndIncludeInRelease
335 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
336 〈\latexrelease〉           {\@centercr}{\Make robust}%
337 〈\latexrelease〉
338 〈\latexrelease〉\def\@centercr{\ifhmode \unskip\else \nolnerr\fi
339 〈\latexrelease〉          \par\@ifstar{\nobreak\@xcentercr}\@xcentercr}
340 〈\latexrelease〉
341 〈\latexrelease〉\EndIncludeInRelease
342 (*2ekernel)

```

(End of definition for \@centercr.)

\@xcentercr

```

343 \def\@xcentercr{\addvspace{-\parskip}\@ifnextchar
344   [\\@icentercr\ignorespaces}

```

(End of definition for \@xcentercr.)

\@icentercr

```

345 〈/2ekernel〉
346 〈*2ekernel | \latexrelease〉
347 〈\latexrelease〉\IncludeInRelease{2020/10/01}%
348 〈\latexrelease〉           {\@icentercr}{centering, etc support calc}%
349 \def\@icentercr[#1]{\@vspace@calcify{#1}\ignorespaces}
350 〈/2ekernel | \latexrelease〉
351 〈\latexrelease〉\EndIncludeInRelease
352 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
353 〈\latexrelease〉           {\@icentercr}{centering, etc support calc}%
354 〈\latexrelease〉
355 〈\latexrelease〉\def\@icentercr[#1]{\vskip #1\ignorespaces}
356 〈\latexrelease〉\EndIncludeInRelease
357 (*2ekernel)

```

(End of definition for \@icentercr.)

center (env.) We use \relax to prevent \item scanning too far.

```

358 \def\centerf{\trivlist \centering\item\relax}
359 \def\endcenter{\endtrivlist}
360 〈/2ekernel〉
361 〈*2ekernel | \latexrelease〉
362 〈\latexrelease〉\IncludeInRelease{2020/10/01}%
363 〈\latexrelease〉           {\centering}{Set finalhyphendemerits}%

```

\centering

```

364 \DeclareRobustCommand\centering{%
365   \let\\@centercr
366   \rightskip\@flushglue\leftskip\@flushglue
367   \finalhyphendemerits=\z@
368   \parindent\z@\parfillskip\z@skip}

```

(End of definition for \centering.)

```

\raggedright
369 \DeclareRobustCommand\raggedright{%
370   \let\\@centercr\@rightsip\@flushglue \rightsip\@rightsip
371   \finalhyphendemerits=\z@
372   \leftskip\z@skip
373   \parindent\z@}

(End of definition for \raggedright.)

\raggedleft
374 \DeclareRobustCommand\raggedleft{%
375   \let\\@centercr
376   \rightsip\z@skip\leftskip\@flushglue
377   \finalhyphendemerits=\z@
378   \parindent\z@\parfillskip\z@skip}

(End of definition for \raggedleft.)

379 </2ekernel | latexrelease>
380 <latexrelease>\EndIncludeInRelease
381 <latexrelease>\IncludeInRelease{2019/10/01}%
382 <latexrelease>          {\centering}{\Make commands robust}%
383 <latexrelease>
384 <latexrelease>\DeclareRobustCommand\centering{%
385 <latexrelease>  \let\\@centercr
386 <latexrelease>  \rightsip\@flushglue\leftskip\@flushglue
387 <latexrelease>  \parindent\z@\parfillskip\z@skip}
388 <latexrelease>\DeclareRobustCommand\raggedright{%
389 <latexrelease>  \let\\@centercr\@rightsip\@flushglue \rightsip\@rightsip
390 <latexrelease>  \leftskip\z@skip
391 <latexrelease>  \parindent\z@}
392 <latexrelease>\DeclareRobustCommand\raggedleft{%
393 <latexrelease>  \let\\@centercr
394 <latexrelease>  \rightsip\z@skip\leftskip\@flushglue
395 <latexrelease>  \parindent\z@\parfillskip\z@skip}
396 <latexrelease>\EndIncludeInRelease
397 <latexrelease>
398 <latexrelease>\IncludeInRelease{0000/00/00}%
399 <latexrelease>          {\centering}{\Make commands robust}%
400 <latexrelease>
401 <latexrelease>\kernel@make@fragile\centering
402 <latexrelease>\kernel@make@fragile\raggedright
403 <latexrelease>\kernel@make@fragile\raggedleft
404 <latexrelease>
405 <latexrelease>\EndIncludeInRelease
406 <*2ekernel>

{@rightsip
407 \newskip{@rightsip \rightsip \z@skip

(End of definition for \@rightsip.)}

```

flushleft (*env.*) We use `\relax` to prevent `\item` scanning too far.

```

408 \def\flushleft{\trivlist \raggedright\item\relax}
409 \def\endflushleft{\endtrivlist}

```

flushright (*env.*) We use `\relax` to prevent `\item` scanning too far.

```
410 \def\flushright{\trivlist \raggedleft\item\relax}
411 \def\endflushright{\endtrivlist}
```

1.3 Verbatim

```
412 \message{verbatim,}
```

The verbatim environment uses the fixed-width `\ttfamily` font, turns blanks into spaces, starts a new line for each carriage return (or sequence of consecutive carriage returns), and interprets *every* character literally. I.e., all special characters `\`, `{`, `$`, etc. are `\catcode`'d to 'other'.

The command `\verb` produces in-line verbatim text, where the argument is delimited by any pair of characters. E.g., `\verb #...#` takes '...' as its argument, and sets it verbatim in `\ttfamily` font.

The `*-variants` of these commands are the same, except that spaces print as the TeXbook's space character instead of as blank spaces.

```
\@vobeyspaces
```

```
413 {\catcode`\\ =\active%
414 \gdef\@vobeyspaces{\catcode`\\ \active\let \xobeysp}}
```

(*End of definition for `\@vobeyspaces`.*)

```
\@xobeysp
```

(*End of definition for `\@xobeysp`.*)

```
\@xverbatim
```

```
\@sxverbatim
415 \begingroup \catcode `|=0 \catcode '['= 1
416 \catcode']=2 \catcode '\{}=12 \catcode '\}=12
417 \catcode`\\"=12 \gdef|@xverbatim#1\end{verbatim}#[#1\end[verbatim]]
418 \gdef|@sxverbatim#1\end{verbatim*}#[#1\end[verbatim*]]
419 |endgroup
```

(*End of definition for `\@xverbatim` and `\@sxverbatim`.*)

\@verbatim Real start of verbatim environment We use `\relax` to prevent `\item` scanning too far.

```
420 </2ekernel>
421 <*2ekernel | latexrelease>
422 <latexrelease>\IncludeInRelease{2017-04-15}{\@verbatim}%
423 <latexrelease>                                {Disable hyphenation in verbatim}%
424 \def\@verbatim{\trivlist \item\relax
425   \if@minipage\else\vskip\parskip\fi
426   \leftskip\@totallleftmargin\rightskip\z@skip
427   \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.

```
428 \@par
```

Set `\language` here to suppress hyphenation. Done this way rather than setting `\hyphenchar` as that is a global setting.

```
429  \language\l@nohyphenation
430  \@tempswafalse
431  \def\part%
432    \if@tempswa
```

A `\leavevmode` added: needed if, for example, a blank verbatim line is the first thing in a list item (wow!).

```
433  \leavevmode \null \@@par\penalty\interlinepenalty
434  \else
435    \atempswatrue
436    \ifhmode\@@par\penalty\interlinepenalty\fi
437  \fi}%
```

To allow customization we hide the font used in a separate macro.

```
438  \let\do\@makeother \dospecials
439  \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`!

```
440  \everypar \expandafter{\the\everypar \unpenalty}%
441  }
442  </2ekernel | latexrelease>
443  <latexrelease>\EndIncludeInRelease
444  <latexrelease>\IncludeInRelease{0000-00-00}{\@verbatim}%
445  <latexrelease>                                {Disable hyphenation in verbatim}%
446  <latexrelease>\def\@verbatim{\trivlist \item\relax
447  <latexrelease> \if@minipage\else\vskip\parskip\fi
448  <latexrelease> \leftskip\@totalleftmargin\rightskip\z@skip
449  <latexrelease> \parindent\z@\parfillskip\@flushglue\parskip\z@skip
450  <latexrelease> \@@par
451  <latexrelease> \atempswafalse
452  <latexrelease> \def\par{%
453  <latexrelease>   \if@tempswa
454  <latexrelease>     \leavevmode \null \@@par\penalty\interlinepenalty
455  <latexrelease>   \else
456  <latexrelease>     \atempswatrue
457  <latexrelease>     \ifhmode\@@par\penalty\interlinepenalty\fi
458  <latexrelease>   \fi}%
459  <latexrelease> \let\do\@makeother \dospecials
460  <latexrelease> \obeylines \verbatim@font \noligs
461  <latexrelease> \hyphenchar\font\m@ne
462  <latexrelease> \everypar \expandafter{\the\everypar \unpenalty}%
463  <latexrelease> }
464  <latexrelease>\EndIncludeInRelease
465  <*2ekernel>
```

(End of definition for `\@verbatim`.)

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

```
466  \def\verb@verbatim@{\@verbatim \frenchspacing\@vobeyspaces \x@verbatim}%
467  \def\endverb@verbatim@{\if@newlist \leavevmode\fi\endtrivlist}
```

(End of definition for `\verb@verbatim@` and `\endverb@verbatim@`.)

`\verbatim@font` Macro to select the font used for verbatim typesetting. It also does other work if necessary for the font used.

```

468 \def\verbatim@font{\normalfont\ttfamily}

(End of definition for \verbatim@font.)

469 </2ekernel>
470 <*2ekernel | latexrelease>
471 <| latexrelease>\IncludeInRelease{2018/12/01}%
472 <| 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

```

473 \DeclareRobustCommand\asciispace{\char 32 }

(End of 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.

```

474 \ifx\Umathcode\@undefined
475   \let\verbvisible\asciispace % Pdftex version
476 \else
477   \DeclareRobustCommand\verbvisible
478     {\leavevmode{\usefont{OT1}{cmtt}{m}{n}\asciispace}} % xetex/luatex version
479 \fi

```

(End of 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.

```

480 \def\@setupverbvisible{%
481   \ifx\verbvisible\asciispace
482     \let\xobeysp\asciispace
483   \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.

```

484   \setbox\z@\hbox{x}%
485   \setbox\@verbvisiblebox\hbox to\wd\z@{\hss\verbvisible\hss}%
486   \def\xobeysp{\leavevmode\copy\@verbvisiblebox}%
487 \fi
488 }

```

(End of definition for \@setupverbvisible.)

\@verbvisiblespacebox The box to hold the visible space character if it isn't in slot 32 in the current typewriter font.

489 \newbox\@verbvisiblespacebox

(End of definition for \@verbvisiblespacebox.)

verbatim* (env.) For verbatim* we also set up the correct visible space character definition and then run \@vobeyspaces. As this code is not called as part of the normal verbatim environment (the method is done the other way around this time) we don't have to check if space is already active—it shouldn't be.

```
490 \@namedef{verbatim*}{\@verbatim
491   \@setupverbvisiblespace
492   \frenchspacing\@vobeyspaces\@sxverbatim}
493 \expandafter\let\csname endverbatim*\endcsname =\endverbatim

494 {/2ekernel | latexrelease}
495 <latexrelease>\EndIncludeInRelease
496 <latexrelease>\IncludeInRelease{0000/00/00}%
497 <latexrelease>          {\verbvisible}{Setup visible space for \verb}%
498 <latexrelease>
499 <latexrelease>\@namedef{verbatim*}{\@verbatim\@sxverbatim}
500 <latexrelease>
501 <latexrelease>\let\asciispace      \@undefined
502 <latexrelease>\let\verbvisiblespace    \@undefined
503 <latexrelease>\let\@setupverbvisiblespace\@undefined
504 <latexrelease>\let\@verbvisiblespacebox \@undefined
505 <latexrelease>\EndIncludeInRelease
506 {*2ekernel}
```

\@sverb Definitions of \@sverb and \@verb changed so \verb+ foo+ does not lose leading blanks when it comes at the beginning of a line. Change made 24 May 89. Suggested by Frank Mittelbach and Rainer Schöpf.

```
507 {/2ekernel}
508 {*2ekernel | latexrelease}
509 <latexrelease>\IncludeInRelease{2020/10/01}%
510 <latexrelease>          {\@sverb}{Drop spaces before \verb delimiter}%
```

If the user 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).

```
511 \def\@sverb#1{\if\noexpand#1 \expandafter\@sverb\else\@@sverb{#1}\fi}

512 \def\@@sverb#1{%
513   \catcode`#1\active
514   \lccode`\~`#1%
515   \gdef\verb@balance@group{\verb@egroup
516     \@latex@error{\noexpand\verb illegal in argument}\@ehc}%
517   \aftergroup\verb@balance@group
518   \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.

```
519  \ifnum\catcode`\\ =\active
520  \else  \@setupverbvisiblespace \@vobeyspaces \fi
521 }

522 </2ekernel | latexrelease>
523 <latexrelease>\EndIncludeInRelease
524 <latexrelease>\IncludeInRelease{2018/12/01}%
525 <latexrelease>          {\@sverb}{Setup visible space for \verb}%
526 <latexrelease>
527 <latexrelease>\def\@sverb#1{%
528 <latexrelease>  \catcode`#1\active
529 <latexrelease>  \lccode`~`#1%
530 <latexrelease>  \gdef\verb@balance@group{\verb@egroup
531 <latexrelease>    \@latex@error{\noexpand\verb illegal in command argument}\@ehc}%
532 <latexrelease>  \aftergroup\verb@balance@group
533 <latexrelease>  \lowercase{\let~\verb@egroup}%
534 <latexrelease>  \ifnum\catcode`\\ =\active
535 <latexrelease>  \else  \@setupverbvisiblespace \@vobeyspaces \fi
536 <latexrelease>
537 <latexrelease>\let\@sverb\@undefined
538 <latexrelease>\EndIncludeInRelease
539 <latexrelease>
540 <latexrelease>\IncludeInRelease{0000/00/00}%
541 <latexrelease>          {\@sverb}{Setup visible space for \verb}%
542 <latexrelease>\def\@sverb#1{%
543 <latexrelease>  \catcode`#1\active
544 <latexrelease>  \lccode`~`#1%
545 <latexrelease>  \gdef\verb@balance@group{\verb@egroup
546 <latexrelease>    \@latex@error{\noexpand\verb illegal in command argument}\@ehc}%
547 <latexrelease>  \aftergroup\verb@balance@group
548 <latexrelease>  \lowercase{\let~\verb@egroup}%
549 <latexrelease>
550 <latexrelease>\EndIncludeInRelease
551 <*2ekernel>
```

(End of definition for `\@sverb` and `\@sverb`.)

`\@makeother`

```
552 \def\@makeother#1{\catcode`#112\relax}
```

(End of definition for `\@makeother`.)

`\verb@balance@group`

```
553 \let\verb@balance@group\@empty
```

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

`\verb@egroup`

```
554 \def\verb@egroup{\global\let\verb@balance@group\@empty\egroup}
```

(End of definition for `\verb@egroup`.)

```
\verb@eol@error
 555 \begingroup
 556   \obeylines%
 557   \gdef\verb@eol@error{\obeylines%
 558     \def^~M{\verb@egroup@\latex@error{%
 559       \noexpand\verb ended by end of line}\@ehc}}%
 560 \endgroup
```

(End of definition for `\verb@eol@error.`)

`\verb` Typesetting a small piece verbatim.

```
561 </2ekernel>
562 <*2ekernel | latexrelease>
563 <latexrelease>\IncludeInRelease[2017-04-15]{\verb}%
564 <latexrelease>                                {Disable hyphenation in verb}%
565 \def\verb{\relax\ifmmode\hbox\else\leavevmode\null\fi
566   \bgroup
567   \verb@eol@error \let\do\@makeother \dospecials
568   \verbatim@font\@noligs}
```

Set `\language` here to suppress hyphenation. Done this way rather than setting `\hyphenchar` as that is a global setting.

```
569   \language\l@nohyphenation
570   \@ifstar\@sverb\@verb}
571 </2ekernel | latexrelease>
572 <latexrelease>\EndIncludeInRelease
573 <latexrelease>\IncludeInRelease[0000-00-00]{\verb}%
574 <latexrelease>                                {Disable hyphenation in verb}%
575 <latexrelease>\def\verb{\relax\ifmmode\hbox\else\leavevmode\null\fi
576 <latexrelease> \bgroup
577 <latexrelease> \verb@eol@error \let\do\@makeother \dospecials
578 <latexrelease> \verbatim@font\@noligs
579 <latexrelease> \@ifstar\@sverb\@verb}
580 <latexrelease>\EndIncludeInRelease
581 <*2ekernel>
```

(End of definition for `\verb.`)

```
\@verb
 582 \def\@verb{\vobeyspaces \frenchspacing \@sverb}
```

(End of definition for `\@verb.`)

`\verbatim@nolig@list`

```
583 \def\verbatim@nolig@list{\do`\'\do`<\do\>\do\,\do`'\do`-`}
```

(End of definition for `\verbatim@nolig@list.`)

`\do@noligs`

```
584 \def\do@noligs#1{%
585   \catcode`#1\active
586   \begingroup
587     \lccode`\~`#1\relax
588     \lowercase{\endgroup\def~{\leavevmode\kern\z@\char`#1}}}
```

(End of definition for \do@noligs.)

\@noligs To stay compatible with packages that use \@noligs we keep it.

589 \def\@noligs{\let\do\do@noligs \verbatim@nolig@list}

(End of definition for \@noligs.)

590 \langle /2ekernel \rangle

File I

ltmath.dtx

1 Math setup

This file contains a lot of the original plain T_EX code, as well as the L^AT_EX environments for math. It still needs sorting out.

```
1  <*2ekernel>
2  \message{math definitions,}
```

1.1 Math commands based on plain T_EX

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 of 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 of definition for `\bmod`)

`\pmod`

```

39 \DeclareRobustCommand\pmod[1]{%
40   \allowbreak\mkern18mu(\operator@font mod}\,,\,#1)}

```

(End of 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 of 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 of definition for `\jot`.)

`\interdisplaylinepenalty`

```

55 \newcount\interdisplaylinepenalty
56 \interdisplaylinepenalty=100

```

(End of definition for `\interdisplaylinepenalty`.)

`\choose`

```

57 \def\choose{\atopwithdelims()}

```

(End of definition for `\choose`.)

`\brack`

```

58 \def\brack{\atopwithdelims[]}

```

(End of definition for `\brack`.)

```

\brace
59 \def\brace{\atopwithdelims\{\}}
(End of 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 of 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 of 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\phantom{\v@true\h@true\ph@nt}

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   \leavevmode@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   \setbox\tw@\null
122   \ifv@ \ht\tw@\ht\z@ \dp\tw@\dp\z@\fi
123   \ifh@ \wd\tw@\wd\z@\fi \box\tw@}
124 <latexrelease>\EndIncludeInRelease
125 <*2ekernel>

```

(End of 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 </2ekernel>
136 <*2ekernel | latexrelease>
137 <latexrelease>\IncludeInRelease{2022/11/01}%
138 <latexrelease>           {\mathsm@sh}{Guard against reboxing}%
139 \def\mathsm@sh#1#2{%
140   \setbox\z@\hbox{$\m@th#1{#2}$}%

```

The empty brace groups in front of the smashed box (which is placed by `\finsm@sh`) ensures that a `\smash` in math is not just producing a single box with its dimensions altered, but a box plus this second ord atom. The reason is that TeX sometimes reboxes

a box if its the only thing in a place like the denominator of a fraction. This would then undo the smashing and the additional ord atom prevents that. Two ord atoms in a row do not alter the horizontal spacing in a formula so this is otherwise transparent.

```

141  {}\\finsm@sh}
142  </2ekernel | latexrelease>
143  <latexrelease>\\EndIncludeInRelease
144  <latexrelease>\\IncludeInRelease{0000/00/00}%
145  <latexrelease>                                {\\mathsm@sh}{Guard against reboxing}%
146  <latexrelease>\\def\\mathsm@sh#1#2{%
147  <latexrelease>  \\setbox\\z@\\hbox{\\m@th#1{#2}}\\finsm@sh}
148  <latexrelease>\\EndIncludeInRelease
149  {*2ekernel}

150 </2ekernel>
151 {*2ekernel | latexrelease}
152 <latexrelease>\\IncludeInRelease{2018/12/01}%
153 <latexrelease>                                {\\finsm@sh}{Start LR-mode}%
154 \\def\\finsm@sh{\\ht\\z@\\z@ \\dp\\z@\\z@ \\leavevmode@ifvmode\\box\\z@}
155 </2ekernel | latexrelease>
156 <latexrelease>\\EndIncludeInRelease
157 <latexrelease>\\IncludeInRelease{0000/00/00}%
158 <latexrelease>                                {\\finsm@sh}{Start LR-mode}%
159 <latexrelease>\\def\\finsm@sh{\\ht\\z@\\z@ \\dp\\z@\\z@ \\box\\z@}
160 <latexrelease>\\EndIncludeInRelease
161 {*2ekernel}

```

(*End of definition for \smash.*)

\buildrel

```
162 \\def\\buildrel#1\\over#2{\\mathrel{\\mathop{\\kern\\z@#2}\\limits^{\\#1}}}
```

(*End of definition for \buildrel.*)

```

163 </2ekernel>
164 {*2ekernel | latexrelease}
165 <latexrelease>\\IncludeInRelease{2019/10/01}%
166 <latexrelease>                                {\\cases}{Make commands robust}%

```

\cases

```

167 \\DeclareRobustCommand*\\cases[1]{\\left\\{\\,\\,\\vcenter{\\normalbaselines\\m@th
168  \\ialign{$##\\hfil&&\\quad##\\hfil\\crcr#1\\crcr}}\\right.\\}

```

(*End of definition for \cases.*)

\matrix

```

169 \\DeclareRobustCommand*\\matrix[1]{\\null\\,\\vcenter{\\normalbaselines\\m@th
170  \\ialign{\\hfil##\\hfil&&\\quad\\hfil##\\hfil\\crcr
171  \\mathstrut\\crcr\\noalign{\\kern-\\baselineskip}
172  #1\\crcr\\mathstrut\\crcr\\noalign{\\kern-\\baselineskip}}}\\,\\}

```

(*End of definition for \matrix.*)

\pmatrix

```
173 \\DeclareRobustCommand*\\pmatrix[1]{\\left(\\matrix{\\right)}}
```

```

(End of definition for \pmatrix.)

174 </2ekernel | latexrelease>
175 <latexrelease>\EndIncludeInRelease
176 <latexrelease>\IncludeInRelease{0000/00/00}%
177 <latexrelease> {\cases}{\Make commands robust}%
178 <latexrelease>
179 <latexrelease>\kernel@make@fragile\cases
180 <latexrelease>\kernel@make@fragile\matrix
181 <latexrelease>\kernel@make@fragile\pmatrix
182 <latexrelease>
183 <latexrelease>\EndIncludeInRelease
184 {*2ekernel}

\bordermatrix
185 \def\bordermatrix#1{\begingroup \m@th
186   \tempdima 8.75\p@
187   \setbox\z@\vbox{%
188     \def\cr{\crcr\noalign{\kern2\p@\global\let\cr\endline}}%
189     \ialign{$##$\hfil\kern2\p@\kern\@tempdima&\thinspace\hfil$##$\hfil
190       \quad\hfil$##$\hfil\crcr
191       \omit\strut\hfil\crcr\noalign{\kern-\baselineskip}%
192       #1\crcr\omit\strut\cr}%
193   \setbox\tw@\vbox{\unvcopy\z@\global\setbox\one\lastbox}%
194   \setbox\tw@\hbox{\unhbox\one\unskip\global\setbox\one\lastbox}%
195   \setbox\tw@\hbox{$\kern\wd\one\kern-\@tempdima\left(\kern-\wd\one
196     \global\setbox\one\vbox{\box\one\kern2\p@}%
197     \vcenter{\kern-\ht\one\unvbox\z@\kern-\baselineskip}\,,\right)$}%
198   \null;\vbox{\kern\ht\one\box\tw@}\endgroup

(End of definition for \bordermatrix.)

\openup
199 \protected\def\openup{\afterassignment\openup\dimen@}
200 \def\openup{\advance\lineskip\dimen@
201   \advance\baselineskip\dimen@
202   \advance\lineskiplimit\dimen@}

(End of definition for \openup.)

\displaylines
203 \newif\ifdt@p
204 \def\displ@y{\global\dt@ptrue\openup\jot\m@th
205   \everycr{\noalign{\ifdt@p \global\dt@pfalse \ifdim\prevdepth>-1000\p@
206     \vskip-\lineskiplimit \vskip\normalineskiplimit \fi
207     \else \penalty\interdisplaylinepenalty \fi}}}
208 \def\@lign{\tabskip\z@skip\everycr{}% restore inside \displ@y
209 \def\displaylines#1{\displ@y \tabskip\z@skip
210   \halign{\hb@xt@{\displaywidth{$\@lign\hfil\displaystyle##\hfil$}}\crcr
211     #1\crcr}}
```

(End of definition for \displaylines.)

```

\sp
\sb  212 \let\sp=^
213 \let\sb=_
```

(End of definition for `\sp` and `\sb`.)

```

\tmspace
\thinspace
\!_
\negthinspace
\:
\medspace
\negmedspace
\;
\thickspace
\negthickspace
```

Originally L^AT_EX 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.

```

214 </2ekernel>
215 <*2ekernel | latexrelease>
216 <latexrelease>\IncludeInRelease{2020/10/01}%
217 <latexrelease>          {\tmspace}{amsmath spacing commands}%

\tmspace is really meant to be an internal command so it doesn't necessarily has to be robust but it was robust in amsmath so we leave it like that.
```

```

218 \DeclareRobustCommand\tmspace[3]{%
219   \ifmmode\mskip#1#2\else\leavevmode@ifvmode\kern#1#3\fi\relax}
```

In `amsmath` the text kern is $.16667\text{em}$. For compatibility reasons we keep the longer one.

```

220 \DeclareRobustCommand\,{\tmspace+\thinmuskip{.16667em}}
221 \let\thinspace\,
222 \DeclareRobustCommand\!{\tmspace-\thinmuskip{.16667em}}
223 \let\negthinspace\!
224 \DeclareRobustCommand\:{\tmspace+\medmuskip{.2222em}}
225 \let\medspace\:
```

L^AT_EX has a second name for this in its manual:

```

226 \let\>=\:
227 \DeclareRobustCommand\negmedspace{\tmspace-\medmuskip{.2222em}}
228 \DeclareRobustCommand\,{\tmspace+\thickmuskip{.2777em}}
229 \let\thickspace\;
230 \DeclareRobustCommand\negthickspace{\tmspace-\thickmuskip{.2777em}}
231 </2ekernel | latexrelease>
232 <latexrelease>\EndIncludeInRelease

233 <latexrelease>\IncludeInRelease{0000/00/00}%
234 <latexrelease>          {\tmspace}{amsmath spacing commands}%
235 <latexrelease>
236 <latexrelease>\let\tmspace\@undefined
237 <latexrelease>\DeclareRobustCommand\,{\,}{%
238 <latexrelease>  \relax\ifmmode\mskip\thinmuskip\else\thinspace\fi}
239 <latexrelease>\DeclareRobustCommand\thinspace{\leavevmode@ifvmode\kern .16667em }
240 <latexrelease>\DeclareRobustCommand\negthinspace{\leavevmode@ifvmode\kern-.16667em }
241 <latexrelease>\def\>{\mskip\medmuskip}
242 <latexrelease>\let\>=\>
243 <latexrelease>\def\,{\mskip\thickmuskip}
244 <latexrelease>\def\!{\mskip-\thinmuskip}
245 <latexrelease>
```

```

246  ⟨latexrelease⟩\let\negmedspace\@undefined
247  ⟨latexrelease⟩\let\negthickspace\@undefined
248  ⟨latexrelease⟩
249  ⟨latexrelease⟩\EndIncludeInRelease
250  {*2ekernel}

(End of definition for \tmspace and others.)

\*
251 \DeclareRobustCommand\*{\discretionary{\thinspace}{\the\textfont2\char2}{}{}}

(End of definition for \*.)

\: Nickname for the medium space since \> is not available inside tabbing.
252 \%{\let\:=\>

(End of definition for \::)

\active@math@prime This is the definition of the active math prime.
253 \def\active@math@prime{^{\bgroup\prim@s} }

(End of definition for \active@math@prime.)

\prime@s
254 {\catcode`'=active \global\let'\active@math@prime}
255 \def\prim@s{%
256   \prime\futurelet\let@token\pr@m@s}
257 \def\pr@m@s{%
258   \ifx'\let@token
259     \expandafter\pr@@s
260   \else
261     \ifx`^{\let@token
262       \expandafter\expandafter\expandafter\pr@@t
263     \else
264       \egroup
265     \fi
266   \fi}
267 \def\pr@@s{\prim@s}
268 \def\pr@@t{\#2\egroup}

(End of definition for \prime@s.)

269 {\catcode`\_=active \gdef_`{\_} } % _ in math is
270                                % either subscript or \

```

1.2 Math Environments

- \(\) Produces \\$...\$ with checks that \(\ isn't used in math mode, and that \) is only used
- \() in math mode begun with \(\.

```

271  </2ekernel>
272  <| latexrelease>\IncludeInRelease{2015/01/01}{\(){\Make \(\ robust}\%
273  {*2ekernel | latexrelease}
274  \DeclareRobustCommand\(\{%
275  \relax\ifmmode\@badmath\else$\fi}\%
276  \DeclareRobustCommand\){%
277  \relax\ifmmode\ifinner$\else\@badmath\fi\else \@badmath\fi}\%
278  </2ekernel | latexrelease>
279  <| latexrelease>\EndIncludeInRelease
280  <| latexrelease>\IncludeInRelease{0000/00/00}{\(){\Make \(\ robust}\%
281  <| latexrelease>\def\(\{%
282  <| latexrelease> \relax\ifmmode\@badmath\else$\fi}\%
283  <| latexrelease>\expandafter\let\csname\string( \endcsname\@undefined
284  <| latexrelease>\def\){%
285  <| latexrelease> \relax\ifmmode\ifinner$\else\@badmath\fi\else \@badmath\fi}\%
286  <| latexrelease>\expandafter\let\csname\string) \endcsname\@undefined
287  <| latexrelease>\EndIncludeInRelease
288  {*2ekernel}

```

(End of definition for \(\ and \).)

- \[Produces \$\$...\$\$ with checks that \[isn't used in math mode, and that \] is only used
- \] in display math mode (though there is no real test that this display math started with \[and not with \$\$).

```

289  </2ekernel>
290  <| latexrelease>\IncludeInRelease{2015/01/01}{\[]{\Make \[ robust}\%
291  {*2ekernel | latexrelease}
292  \DeclareRobustCommand\[{\%
293  \relax\ifmmode
294    \@badmath
295  \else
296    \ifvmode
297      \nointerlineskip
298      \makebox[.6\linewidth]{}}\%
299    \fi
300    $$\%$$ BRACE MATCH HACK
301  \fi
302 }%
303 \DeclareRobustCommand\]{%
304  \relax\ifmmode
305  \ifinner
306    \@badmath
307  \else
308    $$\%$$ BRACE MATCH HACK
309  \fi
310  \else
311    \@badmath
312  \fi
313  \ignorespaces
314 }%

```

```

315  </2ekernel | latexrelease>
316  <latexrelease>\EndIncludeInRelease
317  <latexrelease>\IncludeInRelease{0000/00/00}{\[]}{\Make \[ robust}%
318  <latexrelease>\def\[%
319  <latexrelease>    \relax\ifmmode
320  <latexrelease>        \c@badmath
321  <latexrelease>    \else
322  <latexrelease>        \ifvmode
323  <latexrelease>            \nointerlineskip
324  <latexrelease>            \makebox[.6\linewidth]{}}%
325  <latexrelease>        \fi
326  <latexrelease>        $$$%$$$ BRACE MATCH HACK
327  <latexrelease>    \fi
328  <latexrelease>}%
329  <latexrelease>\expandafter\let\csname\string[ \endcsname\@undefined
330  <latexrelease>\def\[]{%
331  <latexrelease>    \relax\ifmmode
332  <latexrelease>        \ifinner
333  <latexrelease>            \c@badmath
334  <latexrelease>        \else
335  <latexrelease>            $$$%$$$ BRACE MATCH HACK
336  <latexrelease>        \fi
337  <latexrelease>    \else
338  <latexrelease>        \c@badmath
339  <latexrelease>    \fi
340  <latexrelease>    \ignorespaces
341  <latexrelease>}%
342  <latexrelease>\expandafter\let\csname\string] \endcsname\@undefined
343  <latexrelease>\EndIncludeInRelease
344  <*2ekernel>

```

(End of definition for \[and \].)

math (env.) Disguises for \(...\mathord{)} and \(...\mathord{].}

```

displaymath (env.) 345 \let\math=\(
346 \let\endmath=\)
347 \def\displaymath{\[}
348 \def\enddisplaymath{\]} \c@ignoretrue

```

equation (env.) Numbered equations, using the counter \c@equation. Note: The document style must define \theequation etc., and do the appropriate \c@addtoreset. 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.)

```

349 \c@definecounter{equation}
350 \def\equation{\$ \$\refstepcounter{equation}}
351 \def\endequation{\leqno \hbox{\c@eqnnum} \$ \$\c@ignoretrue}

```

(End of 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.

```

352 \def\c@eqnnum{{\normalfont \normalcolor (\theequation)}}

```

(End of definition for \eqnnum.)

\stackrel A disguise for plain T_EX's buildrel.

353 \DeclareRobustCommand\stackrel[2]{\mathrel{\mathop{\#2}\limits^{\#1}}}

(End of definition for \stackrel.)

\frac A disguise for plain T_EX's \over.

354 \DeclareRobustCommand\frac[2]{{\begingroup\over\endgroup\over#2}}

(End of definition for \frac.)

\sqrt Add an optional argument to plain's \sqrt to give the *n*th root of an expression $\sqrt[n]{e}$.

\@sqrt \DeclareRobustCommand\sqrt{\ifnextchar[\@sqrt\sqrtsign{}}

356 \def\@sqrt[#1]{\root #1\of{}}

(End of definition for \sqrt and \@sqrt.)

eqnarray@cnt) Here's the eqnarray environment: Default is for left-hand side of equations to be flushright. To make them flushleft, \let\@eqnse = \hfil.

\@eqpen \newcount\@eqcnt

\@eqnse \newcount\@eqpen

359 \newif\ifeqns\@eqnswtrue

360 \newskip\@centering

361 \@centering = Opt 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.

362 \def\eqnarray{%

363 \stepcounter{equation}%

364 \def\@currentlabel{\p@equation\theequation}%

365 \def\@currentcounter{equation}%

366 \global\@eqnswtrue

367 \m@th

368 \global\@eqcnt\z@

369 \tabskip\@centering

370 \let\\@\eqnccr

371 \$\$\everycr{} \halign to\displaywidth\bgroup

372 \hskip\@centering\\$ \displaystyle\tabskip\z@skip{\#}\\$ \@eqnse

373 &\global\@eqcnt\@ne\hskip \tw@arraycolsep \hfil{\#}\\$ \hfil

374 &\global\@eqcnt\tw@ \hskip \tw@arraycolsep

375 \\$ \displaystyle{\#}\\$ \hfil\tabskip\@centering

376 &\global\@eqcnt\thr@@ \hb@xt@{\z@}\bgroup\hss##\egroup

377 \tabskip\z@skip

378 \cr

379 }

380 \def\endeqnarray{%

381 \@@eqnccr

382 \egroup

383 \global\advance\c@equation\m@ne

384 \$\$\@ignoretrue

385 }

386 \let\@eqnse=\relax

(End of definition for `\@eqcnt` and others.)

`\nonumber` Switches off equation numbering.

387 `\def\nonumber{\global\@eqnswfalse}`

(End of definition for `\nonumber`.)

```
\@eqncr
\@xeqncr 388 \protected\def\@eqncr{%
\@yeqncr 389   {\ifnum0='}\fi
390   \Cifstar{%
391     \global\@eqpen\OM\@yeqncr
392   }{%
393     \global\@eqpen\interdisplaylinepenalty \@yeqncr
394   }%
395 }
396 \def\@yeqncr{\@testopt\@xeqncr\z@skip}

397 </2ekernel>
398 <*2ekernel | latexrelease>
399 <latexrelease>\IncludeInRelease{2020/10/01}%
400 <latexrelease>           {\@\@eqncr}{eqnarray support calc syntax}%
401 \def\@xeqncr[#1]{%
402   \ifnum0='}\fi}%
403   \C@eqncr
404   \noalign{\penalty\@eqpen\vskip\jot\@vspace\@calcify{#1}}%
405 }
406 </2ekernel | latexrelease>
407 <latexrelease>\EndIncludeInRelease
408 <latexrelease>\IncludeInRelease{0000/00/00}%
409 <latexrelease>           {\@\@eqncr}{eqnarray support calc syntax}%
410 <latexrelease>
411 <latexrelease>\def\@xeqncr[#1]{%
412 <latexrelease>   \ifnum0='}\fi}%
413 <latexrelease>   \C@eqncr
414 <latexrelease>   \noalign{\penalty\@eqpen\vskip\jot\vskip #1\relax}%
415 <latexrelease>}
416 <latexrelease>\EndIncludeInRelease
417 <*2ekernel>
```

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

`\@@eqncr`

```
418 \def\@@eqncr{\let\reserved@a\relax
419   \ifcase\@eqcnt \def\reserved@a{& &}\or \def\reserved@a{& &}%
420   \or \def\reserved@a{&}\else
421     \let\reserved@a\empty
422     \C@latex@error{Too many columns in eqnarray environment}\@ehc\fi
423   \reserved@a \if@eqnsw\@eqnnum\stepcounter{equation}\fi
424   \global\@eqnswtrue\global\@eqcnt\z@\cr}
```

(End of definition for `\@@eqncr`.)

`\eqnarray*\@seqncr`) Here's the eqnarray* environment:

```
425 \let\@seqncr=\@eqncr
426 \namedef{eqnarray*}{\protected\def\@eqncr{\nonumber\@seqncr}\eqnarray}
427 \namedef{endeqnarray*}{\nonumber\endeqnarray}
```

(End of definition for \@seqncr.)

`\lefteqn` `\lefteqn{FORMULA}` typesets FORMULA in display math style flushleft in a box of width zero.

```
428 \def\lefteqn#1{\rlap{$\displaystyle #1$}}
(End of 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.

```
429 \DeclareRobustCommand{\ensuremath}{%
430   \ifmmode
431     \expandafter\@firstofone
432   \else
433     \expandafter\@ensuredmath
434   \fi}
```

(End of definition for \ensuremath.)

`\@ensuredmath` The `\relax` stops `\ensuremath{}` starting display math.

```
435 \long\def\@ensuredmath#1{$\relax#1$}
```

(End of definition for \@ensuredmath.)

LuaTeX contains new math primitives to place expression over or under horizontally extensible glyphs. Before LuaTeX 1.14 these did not work correctly with the `\mathstyle` primitive and sometimes did not use cramped style in consistent ways. For newer version, we opt into the corrected behavior.

```
436 \ifx\mathdefaultsmode\@undefined\else
437   \mathdefaultsmode=1
438 \fi
```

`\leqno` Ensure the (deprecated) `$$.. \leqno 1 $$` ignores spaces.

```
439 </2ekernel>
440 <*2ekernel | latexrelease>
441 <latexrelease>\IncludeInRelease{2023/06/01}%
442 <latexrelease>           {\leqno}{add ignorespaces to leqno}%
443 \let\@kernel@eqno\leqno
444 \let\@kernel@leqno\leqno
445 \protected\def\eqno{\@kernel@eqno\aftergroup\ignorespaces}
446 \protected\def\leqno{\@kernel@leqno\aftergroup\ignorespaces}
447 </2ekernel | latexrelease>
448 <latexrelease>\EndIncludeInRelease
449 <latexrelease>\IncludeInRelease{0000/00/00}%
450 <latexrelease>           {\leqno}{add ignorespaces to leqno}%
451 <latexrelease>\let\eqno\@kernel@eqno
452 <latexrelease>\let\leqno\@kernel@leqno
453 <latexrelease>\EndIncludeInRelease
```

(End of definition for `\eqno` and `\leqno`.)

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.

```
454  {*\leqno}
455  \renewcommand{\eqnnum}{\hb@xt@.01\p@{}%
456          \rlap{\normalfont\normalcolor
457          \hskip -\displaywidth(\theequation)}}
458  {/\leqno}
```

(End of 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^AT_EX 2 _{ε} 's displayed math environments.

- `\mathindent` The amount of indentation of the equations is stored in a register.

```
459  {*\fleqn}
460  \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.

```
461  \AtEndOfClass{\mathindent\leftmargini}
```

(End of definition for `\mathindent`.)

`\[` Begin display math;

```
462  \IncludeInRelease{2015/01/01}{}{\Make[]{robust}}%
463  \DeclareRobustCommand{\relax
464          \ifmmode\@badmath
465          \else
466              \begin{trivlist}%
467                  \begin{parpenalty}\predisplaypenalty
468                  \end{parpenalty}\postdisplaypenalty
469                  \item[]\leavevmode
470                  \hb@xt@\linewidth\bgroup \$\m@th\displaystyle %$
471                  \hskip\mathindent\bgroup
472          \fi}
473  \EndIncludeInRelease
474  \IncludeInRelease{0000/00/00}{}{\Make[]{robust}}%
475  \renewcommand{\relax
476          \ifmmode\@badmath
477          \else
478              \begin{trivlist}%
479                  \begin{parpenalty}\predisplaypenalty
480                  \end{parpenalty}\postdisplaypenalty
481                  \item[]\leavevmode
```

```

482           \hb@xt@{\linewidth}{\bgroup $ \m@th \displaystyle %$}
483           \hskip\mathindent\bgroup
484       \fi}
485 \EndIncludeInRelease

```

(End of definition for \L.)

\] end display math;

```

486 \IncludeInRelease{2015/01/01}{\L}{\Make \L robust}%
487 \DeclareRobustCommand{\relax
488     \ifmmode
489         \egroup $\hfil% $
490         \egroup
491         \end{trivlist}%
492     \else \badmath
493     \fi}
494 \EndIncludeInRelease
495 \IncludeInRelease{0000/00/00}{\J}{\Make \J robust}%
496 \renewcommand{\relax
497     \ifmmode
498         \egroup $\hfil% $
499         \egroup
500         \end{trivlist}%
501     \else \badmath
502     \fi}
503 \EndIncludeInRelease

```

(End of definition for \J.)

equation (env.) The equation environment

```

504 \renewenvironment{equation}%
505   {\begin{parpenalty}\predisplaypenalty
506   \end{parpenalty}\postdisplaypenalty
507   \refstepcounter{equation}%
508   \trivlist \item[] \leavevmode
509   \hb@xt@\linewidth{\bgroup $ \m@th \displaystyle %$}
510   \displaystyle
511   \hskip\mathindent}%

```

Ensure that there is at least a space between formula and equation number so that they don't bump in each other.

```

512   {$.hskip .3em minus .3em\hfil % $
513   \displaywidth\linewidth\hbox{@eqnnum}%
514   \egroup
515   \endtrivlist}

```

eqnarray (env.) The eqnarray environment

```

516 \renewenvironment{eqnarray}%
517   \stepcounter{equation}%
518   \def\currentlabel{\p@equation\theequation}%
519   \def\currentcounter{equation}%
520   \global\eqnswtrue\m@th
521   \global\eqcnt\z@
522   \tabskip\mathindent

```

```
523 \let\\=\\eqnrcr  
524 \setlength\abovedisplayskip{\topsep}%  
525 \ifvmode  
526     \addtolength\abovedisplayskip{\partopsep}%  
527 \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.

File J

ltlists.dtx

1 List, and related environments

The generic commands for creating an indented environment – `enumerate`, `itemize`, `quote`, etc – are:

```
\list{\LABEL} {\COMMANDS} ... \endlist
```

which can be invoked by the user as the list environment. The *LABEL* argument specifies item labeling. *COMMANDS* contains commands for changing the horizontal and vertical spacing parameters.

Each item of the environment is begun by the command `\item[ITEMLABEL]` which produces an item labeled by *ITEMLABEL*. If the argument is missing, then the *LABEL* argument of the `\list` command is used as the item label.

The label is formed by putting `\makelabel{\ITEMLABEL}` in an hbox whose width is either its natural width or else `\labelwidth`, whichever is larger. The `\list` command defines `\makelabel` to have the default definition:

```
\makelabel{\ARG} == BEGIN \hfil ARG END
```

which, for a label of width less than `\labelwidth`, puts the label flushright, `\labelsep` to the left of the item's text. However, `\makelabel` can be `\let` to another command by the `\list`'s *COMMANDS* argument.

A `\usecounter{\foo}` command in the second argument causes the counter *foo* to be initialized to zero, and stepped by every `\item` command without an argument. (`\label` commands within the list refer to this counter.)

When you leave a list environment, returning either to an enclosing list or normal text mode, LaTeX begins a new paragraph if and only if you leave a blank line after the `\end` command. This is accomplished by the `\@endparenv` command.

Blank lines are ignored every other reasonable place—i.e.:

- Between the `\begin{list}` and the first `\item`,
- Between the `\item` and the text of that item,
- Between the end of the last item and the `\end{list}`.

For an environment like quotation, in which items are not labeled, the entire environment is a single item. It is defined by letting `\quotation == \list{}{\...}\item\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^ATeX commands can be used to set them. The commands will be executed in vmode if and only if the `\list` was preceded by a `\par` (or something like an `\end{list}`), so the spacing parameters can be set according to whether the list is inside a paragraph or is its own paragraph.

1.2 Vertical Spacing (skips)

\topsep: Space between first item and preceding paragraph.

\partopsep: Extra space added to \topsep when environment starts a new paragraph (is called in vmode).

\itemsep: Space between successive items.

\parsep: Space between paragraphs within an item – the \parskip for this environment.

1.3 Penalties

\@beginparpenalty: put at the beginning of a list

\@endparpenalty: put at end of list

\@itempenalty: put between items.

1.4 Horizontal Spacing (dimens)

\leftmargin: space between left margin of enclosing environment (or of page if top level list) and left margin of this list. Must be nonnegative.

\rightmargin: analogous.

\listparindent: extra indentation at beginning of every paragraph of a list except the one started by the \item command. May be negative! Usually, labeled lists have \listparindent equal to zero.

\itemindent: extra indentation added right BEFORE an item label.

\labelwidth: nominal width of box that contains the label. If the natural width of the label \leq \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^AT_EX 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
  \itemlabel := 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[@itemlabel]
END

\item[LAB] ==

```

```

BEGIN
if @noparitem = true
then @noparitem := false
    % NOTE: then clause hardly every taken,
    % so made a macro \odonoparitem
\box\@labels :=G \hbox{\hskip -\leftmargin
\box\@labels
\hskip \leftmargin }

if @minipage = false then
    \tempskipa := \lastskip
    \vskip -\lastskip
    \vskip \tempskipa + \outerparskip - \parskip
fi
else if @inlabel = true
    then \indent \par % previous item empty.
fi
if hmode then 2 \unskip's
    % To remove any space at end of prev.
    % paragraph that could cause a blank line.
\par
fi
if @newlist = T
    then if @nobreak = T % Kludge if list follows \section
        then \addvspace{\outerparskip - \parskip}
        else \addpenalty{\beginparpenalty}
            \addvspace{\topsep}
            \addvspace{\parskip} %% added 4 Sep 85
        fi
    else \addpenalty{\itempenalty}
        \addvspace{\itemsep}
    fi
    @inlabel :=G true
fi

\everypar{ @minipage :=G F
    @newlist :=G F
    if @inlabel = true
        then @inlabel :=G false
            \hskip -\parindent
            \box\@labels
            \penalty 0
            %% 3 Oct 85 - allow line break here
            \box\@labels :=G null
        fi
    \everypar{} }
@nobreak :=G false
if @noitemarg = true
then @noitemarg := false
if @nmbrlist
then \refstepcounter{\listctr}

```

```

        fi      fi
        \@tempboxa :=L \hbox{\makelabel{LAB}}
        \box\@labels :=G \@labels \hskip \itemindent
                      \hskip - (\labelwidth + \labelsep)
                      if \wd \@tempboxa > \labelwidth
                        then \box\@tempboxa
                        else \hbox to \labelwidth {\unhbox\@tempboxa}
        fi
        \hskip\labelsep
        \ignorespaces                                %% gobble space up to text
END

```

\makelabel{LABEL} == ERROR %% default to catch lonely \item

```

\usecounter{CTR} == BEGIN @nmbrlist :=L true
                      \@listctr == CTR
                      \setcounter{CTR}{0}
END

```

DEFINE \dimen's and \count
End of historical L^AT_EX 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 of definition for \topskip and others.*)

```

\leftmargin
\rightmargin 9 \newdimen\leftmargin
\listparindent 10 \newdimen\rightmargin
\itemindent 11 \newdimen\listparindent
\labelwidth 12 \newdimen\itemindent
\labelsep 13 \newdimen\labelwidth
\@totallleftmargin 14 \newdimen\labelsep
15 \newdimen\linewidth
16 \newdimen\@totallleftmargin \@totallleftmargin=\z@

```

(*End of definition for \leftmargin and others.*)

```

\leftmargini
\leftmarginii
17 \newdimen\leftmargini
18 \newdimen\leftmarginii
19 \newdimen\leftmarginiii
20 \newdimen\leftmarginiv
21 \newdimen\leftmarginv
22 \newdimen\leftmarginvi

(End of definition for \leftmargini and others.)

\@listdepth
\@itempenalty
\@beginparpenalty
23 \newcount\@listdepth \@listdepth=0
24 \newcount\@itempenalty
25 \newcount\@beginparpenalty
26 \newcount\@endparpenalty

(End of definition for \@listdepth and others.)

\@labels
27 \newbox\@labels

(End of definition for \@labels.)

\if@inlabel
\@inlabelfalse
28 \newif\if@inlabel \@inlabelfalse
\@inlabeltrue
(End of definition for \if@inlabel, \@inlabelfalse, and \@inlabeltrue.)

\if@newlist
\@newlistfalse
29 \newif\if@newlist \@newlistfalse
\@newlisttrue
(End of definition for \if@newlist, \@newlistfalse, and \@newlisttrue.)

\if@noperitem
\@noperitemfalse
30 \newif\if@noperitem \@noperitemfalse
\@noperitemtrue
(End of definition for \if@noperitem, \@noperitemfalse, and \@noperitemtrue.)

\if@noparlist
\@noparlistfalse
31 \newif\if@noparlist \@noparlistfalse
\@noparlisttrue
(End of definition for \if@noparlist, \@noparlistfalse, and \@noparlisttrue.)

\if@noitemarg
\@noitemargfalse
32 \newif\if@noitemarg \@noitemargfalse
\@noitemargtrue
(End of definition for \if@noitemarg, \@noitemargfalse, and \@noitemargtrue.)

\if@newlist
\@newlistfalse
33 \newif\if@nmbrlist \@nmbrlistfalse
\@newlisttrue
(End of definition for \if@newlist, \@newlistfalse, and \@newlisttrue.)

```

```

list\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 of definition for \list.)

```

```
\par@deathcycles
 56 \newcount\par@deathcycles
```

(End of 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_EX. 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          {\@cpar}%
83      \fi
84      \else
85          {\@cpar}%
86      \fi}%
87  \global \cnewlisttrue
88  \outerparskip \parskip}

```

(End of definition for `\@trivlist`.)

`trivlistlist`)

```

89  \def\trivlist{%
90  \parsep\parskip
91  \cnbrlistfalse
92  \trivlist
93  \labelwidth\z@
94  \leftmargin\z@
95  \itemindent\z@

```

We initialise `\@itemlabel` so that a `trivlist` with an `\item` not having an optional argument doesn't produce an error message.

```

96  \let\@itemlabel\empty
97  \def\makelabel##1{##1}

```

(End of definition for `\trivlist`.)

`\endlist`

```

98  \def\endlist{%
99  \global\advance\clistdepth\mone
100 \endtrivlist}

```

(End of definition for `\endlist`.)

The definition of `\trivlist` used to be in `ltspacedtx` so that other commands could be ‘let to it’. They now use `\def`.

`\endtrivlist`

```

101 \def\endtrivlist{%
102  \if@inlabel
103      \leavevmode
104      \global \cnotinlabelfalse
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 of 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 <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 <latexrelease>\EndIncludeInRelease
133 <latexrelease>\IncludeInRelease{0000/00/00}{\@doendpe}{clubpenalty fix}%
134 <latexrelease>\def\@doendpe{\@endpetrue
135 <latexrelease>    \def\par{\@restorepar\everypar{}{\par\@endpefalse}\everypar
136 <latexrelease>        \setbox\z@\lastbox}\everypar{}{\@endpefalse}%
137 <latexrelease>\EndIncludeInRelease

```

(End of definition for `\@endparenv` and `\@doendpe`.)

```

\if@endpe
\@endpefalse
138 \newif\if@endpe
139 \@endpefalse

```

(End of definition for `\if@endpe`, `\@endpefalse`, and `\@endpeltrue`.)

```

\@mklab
140 \def\@mklab#1{\hfil #1}
(End of definition for \@mklab.)

\item
141 \def\item{%
142   \cinmatherr\item
143   \@ifnextchar [\@item{\@noitemargtrue \@item[\@itemlabel]}}%
(End of definition for \item.)

\@donoparitem
144 \def\@donoparitem{%
145   \ifnoparitemfalse
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 of definition for \@donoparitem.)

\@item
156 \def\@item[#1]{%
157   \ifnoparitem
158     \@donoparitem
159   \else
160     \ifinlabel
161       \indent \par
162     \fi
163     \ifhmode
164       \unskip\unskip \par
165     \fi
166     \ifnewlist
167       \ifnobreak
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     \ifminipagefalse
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 of definition for \@item.)
```

\makelabel

```

218 \def\makelabel#1{%
219   \@latex@error{Lonely \string\item--perhaps a missing
220   list environment}\@ehc}
221
222 (End of definition for \makelabel.)
```

\@nbitem

```

223 \def\@nbitem{%
224   \@tempskipa\@outerparskip
225   \advance\@tempskipa -\parskip
226   \addvspace\@tempskipa}
```

(End of definition for \@nbitem.)

\usecounter

```

227 \def\usecounter#1{\@nmbrlisttrue\def\@listctr{#1}\setcounter{#1}\z@}
228
229 (End of 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 `\@enumspacing` and `\@enumdepth`.

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

```

\enumerate ==
BEGIN
if \@enumdepth > 3
  then errormessage: "Too deeply nested".
  else \@enumdepth :=L \@enumdepth + 1
```

```

    \c@enumctr :=L eval(enum@\romannumeral\the\c@enumdepth)
    \list{\label(\c@enumctr)}
        {\usecounter{\c@enumctr}
         \makelabel{LABEL} == \hss \llap{LABEL}}
    fi
END

```

\endenumerate == \endlist
End of historical L^AT_EX 2.09 comments.

```

\c@enumdepth
226 \newcount\c@enumdepth \c@enumdepth = 0
(End of definition for \c@enumdepth.)

```

```

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

```

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

```

enumerate (env.)
231 \def\enumerate{%
232   \ifnum \c@enumdepth > \thr@@\c@toodeep\else
233     \advance\c@enumdepth\@ne
234     \edef\c@enumctr{enum\romannumeral\the\c@enumdepth}%
235     \expandafter
236     \list
237       \csname label\c@enumctr\endcsname
238       {\usecounter{\c@enumctr}\def\makelabel##1{\hss\llap{##1}}\%}
239   \fi}
240 \let\endenumerate =\endlist

```

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

```

\itemize ==
BEGIN
  if \c@itemdepth > 3
    then errormessage: 'Too deeply nested'.
    else \c@itemdepth :=L \c@itemdepth + 1
      \c@itemitem == eval(labelitem\romannumeral\the\c@itemdepth)
      \list{\c@nameuse{\c@itemitem}}
            {\makelabel{LABEL} == \hss \llap{LABEL}}
  fi
END

```

\enditemize == \endlist

End of historical L^AT_EX 2.09 comments.

```

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

itemize (*env.*)

```

242 \def\itemize{%
243   \ifnum \@itemdepth >\thr@@\@toodeep\else
244     \advance\@itemdepth\@ne
245     \edef\@itemitem{\labelitem\romannumeral\the\@itemdepth}%
246     \expandafter
247     \list
248       \csname\@itemitem\endcsname
249       {\def\makelabel##1{\hss\llap{##1}}}\%
250   \fi}
251 \let\enditemize =\endlist
252 ⟨/2ekernel⟩
```

File K

ltboxes.dtx

1 L^AT_EX 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_EX box register.
\savebox \savebox{⟨cmd⟩} ... : \cmd is defined to be a T_EX box register, and the '...' are any \makebox arguments. It is like \makebox, except it doesn't produce text but saves the value in \box \cmd.
\sbox \sbox{⟨cmd⟩}{⟨obj⟩} is an efficient abbreviation for
\savebox{⟨cmd⟩}{⟨obj⟩}.
\lrbox (env.) \begin{lrbox}{⟨cmd⟩}{⟨text⟩}\end{lrbox} is equivalent to
\sbox{⟨cmd⟩}{⟨text⟩}
except that any white space at the beginning and end of ⟨text⟩ is ignored.
\framebox \framebox ... : like \makebox, except it puts a 'frame' around the box. The frame is made of lines of thickness \fboxrule, separated by space \fboxsep from the text – except for \framebox(X,Y) ... , where the thickness of the lines is as for the picture environment, and there is no separation added.
\fbox \fbox{⟨obj⟩} is an abbreviation for \framebox{⟨obj⟩}.
\parbox \parbox[⟨pos⟩][⟨height⟩][⟨inner-pos⟩]{⟨width⟩}{⟨text⟩} : Makes a box with \hsize ⟨width⟩, positioned by ⟨pos⟩ as follows: c : \vcenter (placed in \$...\$ if not in math mode) b : \vbox t : \vtop default value is c. Sets \hsize := ⟨width⟩ and calls \parboxrestore, which does the following: Restores the original definitions of:
\par
\
\-
\'
\`
\=

Resets the following parameters:

\parindent	=	opt
\parskip	=	opt
\ linewidth	=	\hsize
\@totalleftmargin	=	opt
\leftskip	=	opt
\rightskip	=	opt
\@rightskip	=	opt
\parfillskip	=	opt plus 1fil
\lineskip	=	\normallineskip
\baselineskip	=	\normalbaselineskip

Calls \sloppy
Note: \arrayparboxrestore same as \parboxrestore but it doesn't restore \\.

minipage (env.) minipage : Similar to \parbox, except it also makes this look like a page by setting \textwidth == \columnwidth == box width
changes footnotes by redefining:
\@mpfn == mpfootnote
\thempfn == \thempfootnote
\@footnotetext == \@mpfootnotetext
resets the following list environment parameters:
\@listdepth == \@mplistdepth
where \@mplistdepth is initialized to zero,
and executes \minipagerestore to allow the document style to reset any other parameters it desires. It sets @minipage true, and resets \everypar to set it false. This switch keeps \addvspace from putting space at the top of a minipage.
Change added 24 May 89: \minipage sets @minipage globally; \endminipage resets it false.

\rule	\rule[<raised>]{<width>}{<height>} : Makes a <width>*<height> rule, raised <raised>.
\underline	\underline{<text>} : Makes an underlined hbox with <text> in it.
\raisebox	\raisebox{<distance>}{<height>}[<depth>]{<box>} : Raises <box> up by <distance> length (down if <distance> negative). Makes T _E X think that the new box extends <height> above the line and <depth> below, for a total vertical length of <height>+<depth>. Default values of <height> & <depth> = actual height and depth of box in new position.

```

1 (*2ekernel)
2 \message{boxes,}

```

\makebox \makebox User level command just looks for optional [or (.

```

3 
```

- 4 (latexrelease)\IncludeInRelease{2015/01/01}%
- 5 (latexrelease) {\makebox}{Make \makebox robust}%
- 6 (*2ekernel | latexrelease)
- 7 \DeclareRobustCommand\makebox{%
- 8 \leavevmode
- 9 \@ifnextchar(%)
- 10 \@makepicbox
- 11 {\@ifnextchar[\@makebox\mbox}%
- 12 (/2ekernel | latexrelease)
- 13 (latexrelease)\EndIncludeInRelease
- 14 (latexrelease)\IncludeInRelease{0000/00/00}%
- 15 (latexrelease) {\makebox}{Make \makebox robust}%

```

16  <latexrelease>\def\makebox{%
17  <latexrelease>  \leavevmode
18  <latexrelease>  \@ifnextchar(%)
19  <latexrelease>    \makepicbox
20  <latexrelease>    {\ifnextchar[\@makebox\mbox}%
21  <latexrelease>\expandafter\let\csname makebox \endcsname\undefined
22  <latexrelease>\EndIncludeInRelease
23  {*2ekernel}

```

(End of definition for `\makebox`.)

`\mbox` The basic horizontal box command for LATEX.

```
24  \DeclareRobustCommand\mbox[1]{\leavevmode\hbox{\#1}}
```

(End of 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 of 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 of 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 of 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 of 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 of definition for \@imakepicbox.)

\@makepicbox Picture mode form of \makebox.

```
46 \def\@makepicbox(#1,#2){%
47   \@ifnextchar[{\@imakepicbox(#1,#2)}{\@imakepicbox(#1,#2)[ ]}}
```

(End of definition for \@makepicbox.)

\@imakepicbox picture mode version

```
48 (/2ekernel)
49 (*2ekernel | latexrelease)
50 (latexrelease)\IncludeInRelease{2020/10/01}%
51 (latexrelease)           {\@imakepicbox}{default units}%
52 \long\def\@imakepicbox(#1,#2)[#3]#4{%
53   \@defaultunitsset\@tempdimc{#2}\unitlength
54   \vbox to\@tempdimc
55     {\let\mb@b\vss \let\mb@l\hss\let\mb@r\hss
56      \let\mb@t\vss
57      \otfor\reserved@a :=#3\do{%
58        \if s\reserved@a
59          \let\mb@l\relax\let\mb@r\relax
60        \else
61          \expandafter\let\csname mb@\reserved@a\endcsname\relax
62        \fi}%
63      \mb@t
64      \@defaultunitsset\@tempdimc{#1}\unitlength
65      \hb@xt@{\@tempdimc{\mb@l #4\mb@r}}%
66      \mb@b}
```

This kern ensures that a b option aligns on the bottom of the text rather than the baseline. this is the documented behaviour in the L^AT_EX Book. The kern is removed in compatibility mode.

```
67 \kern\z@}
68 (/2ekernel | latexrelease)

69 (latexrelease)\EndIncludeInRelease
70 (latexrelease)\IncludeInRelease{0000/00/00}%
71 (latexrelease)           {\@imakepicbox}{default units}%
72 (latexrelease)\long\def\@imakepicbox(#1,#2)[#3]#4{%
73   (latexrelease) \vbox to#2\unitlength
74   (latexrelease) {\let\mb@b\vss \let\mb@l\hss\let\mb@r\hss
75   (latexrelease) \let\mb@t\vss
76   (latexrelease) \otfor\reserved@a :=#3\do{%
77     (latexrelease) \if s\reserved@a
78       (latexrelease) \let\mb@l\relax\let\mb@r\relax
79     (latexrelease) \else
80       (latexrelease) \expandafter\let\csname mb@\reserved@a\endcsname\relax
81     (latexrelease) \fi}%
82   (latexrelease) \mb@t
83   (latexrelease) \hb@xt@ #1\unitlength{\mb@l #4\mb@r}%
84   (latexrelease) \mb@b
85   (latexrelease) \kern\z@}
86 (latexrelease)\EndIncludeInRelease
87 (*2ekernel)
```

(End of 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 of 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 of definition for \color@begingroup and others.)

\newsavebox Allocate a new ‘savebox’.

115 \def\newsavebox#1{\@if definable{#1}{\newbox#1}}

(End of 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 of definition for `\sbox`.)

`\sbox` Save #1 in a box register.

```

133  \DeclareRobustCommand\sbox[2]{\setbox#1\hbox{%
134    \color@setgroup#2\color@endgroup}}

```

(End of 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 of definition for `\@c savebox`.)

`\@c isavebox`

```

137  \long\def\@c isavebox#1[#2][#3][#4]{%
138    \sbox#1{\@imakebox[#2][#3][#4]}}

```

(End of 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 of 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 of definition for `\@c isavepicbox`.)

`\lrbox` `lrbox`: the new environment form of `\sbox`. Use `\aftergroup` tricks to enable a *local* assignment to be made to the box, in a way that it still has an effect *outside* the `lrbox` environment.

```

144  \def\lrbox#1{%
145    \edef\reserved@a{%
146      \endgroup
147      \setbox#1\hbox{%
148        \begingroup\aftergroup}%
149        \def\noexpand\@currenvir{\@currenvir}%
150        \def\noexpand\@currenvline{\on@line}}%

```

```

151   \reserved@a
152     \endpfalse
153   \color@setgroup
154     \ignorespaces}

(End of definition for \lrbox.)

\endlrbox End the lrbox environment.
155 \def\endlrbox{\unskip\color@endgroup}

(End of definition for \endlrbox.)

\usebox unchanged
156 \DeclareRobustCommand\usebox[1]{\leavevmode\copy #1\relax}

(End of 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 of definition for \fbox.)

\fboxrule user level parameters,
\fboxsep 171 \newdimen\fboxrule
172 \newdimen\fboxsep

(End of 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 of 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 of definition for \framebox.)

\@framebox Deal with optional arguments.

```

197  \def\@framebox[#1]{%
198    \@ifnextchar[%]
199      {\@iframebox[#1]}%
200      {\@iframebox[#1][c]}}

```

(End of definition for \@framebox.)

\@iframebox The handling the optional arguments. In order to set the whole box, including the frame to the specified dimension, we first determine that dimension from the natural size of the text, #3. calculated width.

```

201  \long\def\@iframebox[#1][#2]{#3}%
202  \leavevmode
203  \begin{tempboxa}\hbox{#3}%
204  \setlength\tempdima{#1}%
205  \setbox\tempboxa\hb@xt@\tempdima
206  {\kern\fboxsep\csname bm@#2\endcsname\kern\fboxsep}%
207  \framebox{\kern\fboxrule}%
208  \end{tempboxa}

```

(End of definition for \@iframebox.)

\@frameboxx Common part of \framebox and \fbox. #1 is a negative kern in the \framebox case so that the vertical rules do not add to the width of the box.

```

209  \def\@framebox#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 of definition for `\@frameb@x`.)

`\@framepicbox` Picture mode version.

```

230 \def\@framepicbox(#1,#2){%
231   \@ifnextchar[{\@ifframepicbox(#1,#2)}{\@ifframepicbox(#1,#2)[[]]}}
```

(End of definition for `\@framepicbox`.)

`\@ifframepicbox` Picture mode version.

```

232 \long\def\@ifframepicbox(#1,#2)[#3]{#4}{%
233   \frame{\@imakepicbox(#1,#2)[#3]{#4}}}
```

(End of definition for `\@ifframepicbox`.)

`\parbox` The main vertical-box command for L^AT_EX.

```

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 of definition for `\parbox`.)

`\c@iparbox` Optional argument handling.

```

253 \def\@iparbox[#1]{%
254   \@ifnextchar[%]
255     {\c@iiparbox{#1}}%
256     {\c@iiparbox{#1}\relax[s]}}
```

(End of definition for \@iparbox.)

\@iiparbox Optional argument handling.

```
257 \def\@iiparbox#1[#2]{%
258   \@ifnextchar[%]
259     {\@iiparbox{#1}{#2}}%
260     {\@iiparbox{#1}{#2}[#1]}}
```

(End of 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#3\endcsname}%
278   \if\pboxsw \m@th\fi
279 } \end{tempboxa}
```

(End of 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 \set@nobreak; otherwise this command will be redundant.

```
280 </2ekernel>
281 <|latexrelease|\IncludeInRelease{2017-04-15}%
282 <|latexrelease|           {\normallineskiplimit}%
283 <|latexrelease|           {reset \lineskiplimit}%
284 <*2ekernel | latexrelease>
285 \def\@arrayparboxrestore{%
286   \let\if@nobreak\iffalse
287   \let\ifnoskipsec\iffalse
288   \let\par\@@par
289   \let\-\dischyp
```

Redefined accents to allow changes in font encoding

```
290 \let\@\@acci\let`\@accii\let\=\@acciii
291 \parindent\z@ \parskip\z@skip
292 \everypar{}%
293 \linewidth\hsize
294 \@totalleftmargin\z@
295 \leftskip\z@skip \rightskip\z@skip \@rightskip\z@skip
296 \parfillskip\@flushglue
297 \lineskip\normallineskip
298 \lineskiplimit\normallineskiplimit
299 \baselineskip\normalbaselineskip
300 \sloppy}
301 
```

302 ⟨*latexrelease303 ⟨*latexreleaselatexreleaselatexreleaselatexreleaselatexreleaselatexreleaselatexreleaselatexreleaselatexreleaselatexreleaselatexreleaselatexreleaselatexreleaselatexreleaselatexreleaselatexreleaselatexreleaselatexrelease**

(*End of definition for \@arrayparboxrestore.*)

\@parboxrestore Restore various paragraph parameters, and also \\.

```
322 \def\@parboxrestore{\@arrayparboxrestore\let\\\\@normalcr}
```

(*End of 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 \qquad\@minipagefalse
```

(*End of definition for \if@minipage.*)

\if@in@minipage@env

```
326 \newif\if@in@minipage@env
```

(*End of definition for \if@in@minipage@env.*)

`\minipage` Essentially an environment form of `\parbox`.

```
327 \def\minipage{%
328   \@ifnextchar[%]
329     \@iminiplate
330     {\@iiminipage c\relax[s]}}
```

(End of definition for `\minipage`.)

`\@iminiplate` Optional argument handling.

```
331 \def\@iminiplate[#1]{%
332   \ifnextchar[%]
333     {\@iiminipage{#1}}%
334     {\@iiminipage{#1}\relax[s]}}
```

(End of definition for `\@iminiplate`.)

`\@iiminipage` Optional argument handling.

```
335 \def\@iiminipage#1[#2]{%
336   \ifnextchar[%]
337     {\@iiminipage{#1}{#2}}%
338     {\@iiminipage{#1}{#2}[#1]}}
```

(End of definition for `\@iiminipage`.)

`\@iiiminipage` Internal form of `minipage`.

```
339 \def\@iiiminipage#1#2[#3]{%
340   \leavevmode
341   \pboxswfalse
342   \setlength{\tempdima}{#4}%
343   \def\@mpargs{{#1}{#2}[#3]{#4}}%
344   \setbox\tempboxa\vbox\bgrouptempdima
345   \color\begin{group}
346   \hsize\tempdima
347   \textwidth\hsize \columnwidth\hsize}
```

We check for nested minipages inside the box so that there is always a group resetting the switch even if the code does not use `\begin{group}` to start the minipage.

```
348 \if@in@minipage@env
```

We only issue a warning if the outer minipage contained footnotes because that is the problematical case.

```
349   \ifvoid\@mpfootins\else
350     \@latex@warning{Nested minipage:
351       footnotes may be misplaced}%
352     \fi
353   \else
354     \@in@minipage@envtrue
355   \fi
356   \parboxrestore
357   \def\@mpfn{\mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@
358   \let\@footnotetext\mpfootnotetext
359   \let\@listdepth\mplistdepth \c@mplistdepth\z@
360   \minipagerestore
361   \setminipage{}
```

(End of definition for \@iiiminipage.)

\@minipagerestore Hook so that other styles can reset other commands in a minipage.

362 \let\@minipagerestore=\relax

(End of definition for \@minipagerestore.)

\endminipage

```
363 \def\endminipage{%
364   \par
365   \unskip
366   \ifvoid\@mpfootins\else
367     \vskip\skip\@mpfootins
368     \normalcolor
369     \footnoterule
370     \unvbox\@mpfootins
371   \fi
372   \@minipagetrue %% added 24 May 89
373   \color@endgroup
374   \egroup
375   \expandafter\@iiparbox\@mpargs{\unvbox\@tempboxa}}
```

(End of definition for \endminipage.)

\@mplistdepth Versions of \@listdepth and \footins local to minipage.

\@mpfootins
376 \newcount\@mplistdepth
377 \newinsert\@mpfootins

(End of definition for \@mplistdepth and \@mpfootins.)

\@mpfootnotetext Minipage version of \@footnotetext.

Final \strut added 27 Mar 89, on suggestion by Don Hosek

```
378 </2ekernel>
379 <*2ekernel | latexrelease>
380 <| latexrelease>\IncludeInRelease{2021/11/15}%
381 <| latexrelease>           {\@mpfootnotetext}{footnotetext tagging}%
382 \long\def\@mpfootnotetext#1{%
383   \global\setbox\@mpfootins\vbox{%
384     \unvbox\@mpfootins
385     \reset@font\footnotesize
386     \hsize\columnwidth
387     \parboxrestore
388     \def\@currentcounter{\mpfootnote}%
389     \protected@edef\@currentlabel
390       {\csname p@mpfootnote\endcsname\@thefnmark}%
391     \color@begingroup
392       \makefntext{%
393         \rule{z@\footnotesep}{\ignorespaces#1\finalstrut\strutbox}%
394       }
395     \color@endgroup}
396 </2ekernel | latexrelease>
397 <| latexrelease>\EndIncludeInRelease
```

```

398 〈latexrelease〉\IncludeInRelease{2021/06/01}%
399 〈latexrelease〉                      {\@mpfootnotetext}{footnotetext tagging}%
400 〈latexrelease〉\long\def\@mpfootnotetext#1{%
401 〈latexrelease〉  \global\setbox\@mpfootins\vbox{%
402 〈latexrelease〉    \unvbox\@mpfootins
403 〈latexrelease〉    \reset@font\footnotesize
404 〈latexrelease〉    \hsize\columnwidth
405 〈latexrelease〉    \parboxrestore
406 〈latexrelease〉    \protected@edef\@currentlabel
407 〈latexrelease〉      {\csname p@mpfootnote\endcsname\@thefnmark}%
408 〈latexrelease〉    \color@begingroup
409 〈latexrelease〉      \makefntext{%
410 〈latexrelease〉        \rule{z@}\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
411 〈latexrelease〉    \par
412 〈latexrelease〉    \color@endgroup}%
413 〈latexrelease〉\EndIncludeInRelease
414 〈latexrelease〉\IncludeInRelease{0000/00/00}%
415 〈latexrelease〉                      {\@mpfootnotetext}{footnotetext tagging}%
416 〈latexrelease〉
417 〈latexrelease〉\long\def\@mpfootnotetext#1{%
418 〈latexrelease〉  \global\setbox\@mpfootins\vbox{%
419 〈latexrelease〉    \unvbox\@mpfootins
420 〈latexrelease〉    \reset@font\footnotesize
421 〈latexrelease〉    \hsize\columnwidth
422 〈latexrelease〉    \parboxrestore
423 〈latexrelease〉    \protected@edef\@currentlabel
424 〈latexrelease〉      {\csname p@mpfootnote\endcsname\@thefnmark}%
425 〈latexrelease〉    \color@begingroup
426 〈latexrelease〉      \makefntext{%
427 〈latexrelease〉        \rule{z@}\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
428 〈latexrelease〉    \color@endgroup}%
429 〈latexrelease〉
430 〈latexrelease〉\EndIncludeInRelease
431 〈*2ekernel〉

```

(End of definition for `\@mpfootnotetext`.)

```
432 \newif\if@pboxsw
```

\rule Draw a rule of the specified size.

```

433 〈/2ekernel〉
434 〈latexrelease〉\IncludeInRelease{2015/01/01}%
435 〈latexrelease〉                      {\rule}{\Make \rule robust}%
436 〈*2ekernel | latexrelease〉
437 \DeclareRobustCommand\rule{\@ifnextchar[\@rule{\@rule[\z@]}}%
438 〈/2ekernel | latexrelease〉
439 〈latexrelease〉\EndIncludeInRelease
440 〈latexrelease〉\IncludeInRelease{0000/00/00}%
441 〈latexrelease〉                      {\rule}{\Make \rule robust}%
442 〈latexrelease〉\def\@rule{\@ifnextchar[\@rule{\@rule[\z@]}}%
443 〈latexrelease〉\expandafter\let\csname rule \endcsname\@undefined
444 〈latexrelease〉\EndIncludeInRelease
445 〈*2ekernel〉

```

(End of definition for `\rule`.)

\@rule Internal form of \rule.

```
446  \def\@rule[#1]#2#3{%
447    \leavevmode
448    \hbox{%
449      \setlength\@tempdima{#1}%
450      \setlength\@tempdimb{#2}%
451      \setlength\@tempdimc{#3}%
452      \advance\@tempdimc\@tempdima
453      \vrule\@width\@tempdimb\@height\@tempdimc\@depth-\@tempdima}}

```

(End of definition for \@rule.)

\@@underline Saved primitive \underline.

```
454  \let\@@underline\underline
```

(End of definition for \@@underline.)

\underline L^AT_EX version works outside math.

```
455  \DeclareRobustCommand\underline[1]{%
456    \relax
457    \ifmmode\@@underline{#1}%
458    \else $\@@underline{\hbox{#1}}\m@th$\relax\fi}
```

(End of definition for \underline.)

\raisebox Raise a box, and change its vertical dimensions.

```
459  </2ekernel>
460  <latexrelease>\IncludeInRelease{2015/01/01}%
461  <latexrelease>          {\raisebox}{\Make \raisebox robust}%
462  <*2ekernel | latexrelease>
463  \DeclareRobustCommand\raisebox[1]{%
464    \leavevmode
465    \@ifnextchar[\{\@rsbox{#1}\}\{@irsbox{#1}[]\}]
466  </2ekernel | latexrelease>
467  <latexrelease>\EndIncludeInRelease
468  <latexrelease>\IncludeInRelease{0000/00/00}%
469  <latexrelease>          {\raisebox}{\Make \raisebox robust}%
470  <latexrelease>\def\raisebox#1{%
471  <latexrelease>  \leavevmode
472  <latexrelease>  \@ifnextchar[\{\@rsbox{#1}\}\{@irsbox{#1}[]\}]
473  <latexrelease>\expandafter\let\csname raisebox \endcsname\@undefined
474  <latexrelease>\EndIncludeInRelease
475  <*2ekernel>
```

(End of definition for \raisebox.)

\@rsbox Optional argument handling.

```
476  \def\@rsbox#1[#2]{%
477    \@ifnextchar[\{\@iirsbox{#1}[#2]\}\{@irsbox{#1}[#2]\}}
```

(End of definition for \@rsbox.)

\@argsbox ...

(End of definition for \@argsbox.)

\@irsbox Internal version of \raisebox (less than two optional args).

```
478 \long\def\@irsbox#1[#2]#3{%
479   \begin{tempboxa}\hbox{#3}%
480     \setlength{\tempdima{#1}}%
481     \ifx\#2\else\setlength{\tempdimb{#2}}\fi
482     \setbox\@tempboxa\hbox{\raise\tempdima\box\@tempboxa}%
483     \ifx\#2\else\ht\@tempboxa\tempdimb\fi
484     \box\@tempboxa
485   \end{tempboxa}
```

(End of definition for \@irsbox.)

\@iirsbox Internal version of \raisebox (two optional args).

```
486 \long\def\@iirsbox#1[#2][#3]{%
487   \begin{tempboxa}\hbox{#4}%
488     \setlength{\tempdima{#1}}%
489     \setlength{\tempdimb{#2}}%
490     \setlength{\dimen@{#3}}%
491     \setbox\@tempboxa\hbox{\raise\tempdima\box\@tempboxa}%
492     \ht\@tempboxa\tempdimb
493     \dp\@tempboxa\dimen@
494     \box\@tempboxa
495   \end{tempboxa}
```

(End of 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.

```
496 \def\@finalstrut#1{%
497   \unskip\ifhmode\nobreak\fi\vrule\@width\z@\@height\z@\@depth\dp#1}
```

(End of definition for \@finalstrut.)

1.1 Some low-level constructs

The following commands are basically inherited from plain T_EX.

\leftline These macros place text on a full line either centred or left or right adjusted.
\rightline
\centerline
\@oline

```
498 \def\@oline{\hb@xt@\hsize}
499 \ DeclareRobustCommand\leftline[1]{\@oline{#1\hss}}
500 \ DeclareRobustCommand\rightline[1]{\@oline{\hss#1}}
501 \ DeclareRobustCommand\centerline[1]{\@oline{\hss#1\hss}}
```

(End of definition for \leftline and others.)

`\rlap` These macros place text to the left or right of the current reference point without taking
`\llap` up space.

`\clap`

```
502 \DeclareRobustCommand\rlap[1]{\hb@xt@z@{\#1\hss}}
503 \DeclareRobustCommand\llap[1]{\hb@xt@z@{\hss\#1}}
```

And here is the version that centers, it was initially introduced by `mathtools`.

```
504 \DeclareRobustCommand\clap[1]{\hb@xt@z@{\hss\#1\hss}}
```

(*End of definition for `\rlap`, `\llap`, and `\clap`.*)

```
505 </2ekernel>
```

File L

lttab.dtx

1 Tabbing, Tabular and Array Environments

This section deals with ‘Lining It Up in Columns’. First the `tabbing` environment is defined, and then in second part, `tabular` together with its variants, `tabular*` and `array`.

Note that the `tabular` defined here is essentially the original L^AT_EX 2.09 version, not the extended version described in *The L^AT_EX Companion*. Use the `array` package to obtain the extended version.

1.1 tabbing

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

```
\dimen(\@firsttab + i) = distance of tab stop i from left margin
  0 <= i <= 15 (?).

\dimen\@firsttab is initialized to \totallmargin, so it starts
  at the prevailing left margin.

\@maxtab      = number of highest defined tab register
  probably = \@firsttab + 12
\@nxttabmar = tab stop number of next line's left margin
\@curtabmar = tab stop number of current line's left margin
\@curtab    = number of the current tab. At start of line,
  it equals \@curtabmar
\@hightab   = largest tab number currently defined.
\@tabpush   = depth of \pushtab's

\box\@curline     = contents of current line, excluding left margin
  skip, and excluding contents of current field
\box\@curfield   = contents of current field

@rjfield        = switch: T iff the last field of the line should
  be right-justified at the right margin.

\tabbingsep      = distance left by the \` command between the
  current position and the field that is
  “left-shifted”.

UTILITY MACROS
\@stopfield : closes the current field
\@addfield  : adds the current field to the current line.
\@contfield : continues the current field
\@startfield: begins the next field
\@stopline   : closes the current line and outputs it
```

```

\@startline : starts the next line
\@ifatmargin : an \if that is true iff the current line.
                 has width zero

\@startline ==
BEGIN
  \@curtabmar :=G \@nxttabmar
  \@curtab :=G \@curtabmar
  \box\@curline :=G null
  \@startfield
  \strut
END

\@stopline ==
BEGIN
  \unskip
  \@stopfield
  if @rjfield = T
    then @rjfield :=G F
      \tempdima := \totallmargin + \ linewidth
      \hbox@xt@ \tempdima{\itemfudge
        \hskip \dimen\@curtabmar
        \box\@curline
        \hfil
        \box\@curfield}
    else \addfield
      \hbox {\itemfudge
        \hskip \dimen\@curtabmar
        \box\@curline}
  fi
END

\@startfield ==
BEGIN
  \box\@curfield :=G \hbox {
END

\@stopfield ==
BEGIN
  }
END

\@contfield ==
BEGIN
  \box\@curfield :=G \hbox { \unhbox\@currfield %%} brace matching
END
\@addfield ==
BEGIN
  \box\@curline :=G \unbox\@curline * \unbox\@curfield
END

```

```

\@ifatmargin ==
BEGIN
  if dim of box\@curline = 0pt then
END

\tabbing ==
BEGIN
  \lineskip :=L 0pt
  \> == \@rtab
  \< == \@ltab
  \= == \@settab
  \+ == \@tabplus
  \- == \@tabminus
  \` == \@tabrj
  \' == \@tablab
  \\ == BEGIN \@stopline \@startline END
  \\[DIST] == BEGIN
    \@stopline \vskip DIST \@startline\ignorespaces END
  \\* == BEGIN \@stopline \penalty 10000 \@startline END
  \\*[DIST] == BEGIN \@stopline \penalty 10000 \vskip DIST
    \@startline\ignorespaces END
  \@heighttab := \@nxttabmar :=G \@firsttab
  \@tabpush :=G 0
  \dimen\@firsttab := \@totallleftmargin
  @rjfield :=G F
  \trivlist \item\relax
  if @minipage = F then \vskip \parskip fi
  \box\@tabbbox = \rlap{\indent\the\everypar}
    % note: \the\everypar sets @inlabel :=G F
  \@itemfudge == BEGIN \box\@tabbbox END
  \@startline
  \ignorespaces
END

\@endtabbing ==
BEGIN
  \@stopline
  if \@tabpush > 0 then error message: "unmatched \poptabs" fi
  \endtrivlist
END

\@rtab ==
BEGIN
  \@stopfield
  \@addfield
  if \@curtab < \@heighttab
    then \@curtab :=G \@curtab + 1
    else error message "Undefined Tab" fi

```

```

\@tempdima := \dimen\@curtab - \dimen\@curtabmar
              - width of box \@curline
\box\@curline :=G \hbox{\unhbox\@curline + \hskip\@tempdima}
\@startfield
END

\@settab ==
BEGIN
\@stopfield
\@addfield
if \@curtab < \@maxtab
  then \@curtab :=G \@curtab+1
  else error message: "Too many tabs"    fi
if \@curtab > \@hightab
  then \@hightab :=L \@curtab    fi
\dimen\@curtab :=L \dimen\@curtabmar + width of \box\@curline
\@startfield
END

\@ltab ==
BEGIN
\@ifatmargin
  then if \@curtabmar > \@firsttab
    then \@curtab :=G \@curtab - 1
        \@curtabmar :=G \@curtabmar - 1
    else error message "Too many untabs"    fi
  else error message "Left tab in middle of line"
  fi
END

\@tabplus ==
BEGIN
if \@nxttabmar < \@hightab
  then \@nxttabmar :=G \@nxttabmar+1
  else error message "Undefined tab"
fi
END

\@tabminus ==
BEGIN
if \@nxttabmar > \@firsttab
  then \@nxttabmar :=G \@nxttabmar-1
  else error message "Too many untabs"
fi
END

\@tabrj ==
BEGIN \@stopfield
\@addfield
@rjfield :=G T

```

```

    \@startfield
END

\@tablab ==
BEGIN \@stopfield
    \box\@curline G:= \hbox{\box\@curline %% 'G' added 17 Jun 86
                                \hskip - width of \box\@curfield
                                \hskip -\tabbingsep
                                \box\@curfield
                                \hskip \tabbingsep }

    \@startfield
END

\pushtabs ==
BEGIN
    \@stopfield
    \tabpush :=G \tabpush + 1
    \begingroup
    \@contfield
END

\poptabs ==
BEGIN
    \@stopfield
    if \tabpush > 0
        then \endgroup
            \tabpush :=G \tabpush - 1
        else error message: "Too many \poptabs"
    fi
    \@contfield
END

```

End of historical L^AT_EX 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 of 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 of 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 of definition for `\@nxttabmar` and others.)

```
\@curline  
\@curfield 16 \newbox\@curline  
\@tabbbox 17 \newbox\@curfield  
18 \newbox\@tabbbox
```

(End of definition for `\@curline`, `\@curfield`, and `\@tabbbox`.)

```
\if@rjfield  
19 \newif\if@rjfield
```

(End of 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 of 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 of definition for \@stopline.)

\@startfield

```
46 \def\@startfield{%
47   \global\setbox\@curfield\hbox\bgroup\color@begingroup}
```

(End of definition for \@startfield.)

\@stopfield

```
48 \def\@stopfield{%
49   \color@endgroup\egroup}
```

(End of definition for \@stopfield.)

\@contfield

```
50 \def\@contfield{%
51   \global\setbox\@curfield\hbox\bgroup\color@begingroup
52   \unhbox\@curfield}
```

(End of definition for \@contfield.)

\@addfield

```
53 \def\@addfield{\global\setbox\@curline\hbox{\unhbox
54   \@curline\unhbox\@curfield}}
```

(End of definition for \@addfield.)

\@ifatmargin

```
55 \def\@ifatmargin{\ifdim \wd\@curline =\z@}
```

(End of definition for \@ifatmargin.)

\@tabcr

```
56 \protected\def\@tabcr{\@stopline \@ifstar{\penalty \OM \@xtabcr}\@xtabcr}
```

(End of definition for \@tabcr.)

\@xtabcr

```
57 \def\@xtabcr{\@ifnextchar[\@itabcr{\@startline\ignorespaces}}
```

(End of definition for \@xtabcr.)

\@itabcr

```
58 </2ekernel>
59 <*2ekernel | latexrelease>
60 <latexrelease>\IncludeInRelease{2020/10/01}%
61 <latexrelease>           {\@itabcr}{Tabbing calc syntax}%
62 \def\@itabcr[#1]{\vspace{\calcify{#1}}\@startline\ignorespaces}
63 </2ekernel | latexrelease>
64 <latexrelease>\EndIncludeInRelease
65 <latexrelease>\IncludeInRelease{0000/00/00}%
66 <latexrelease>           {\@itabcr}{Tabbing calc syntax}%
67 <latexrelease>
68 <latexrelease>\def\@itabcr[#1]{\vskip #1\@startline\ignorespaces}
69 <latexrelease>\EndIncludeInRelease
70 <*2ekernel>
```

```

tabbing (env.) We use \relax to prevent \item from scanning too far.

\tabbing 71 \def\tabbing{\lineskip \z@skip\let\>\@rtab\let\<\@ltab\let\=\@settab
72   \let\+\@tabplus\let\-\@tabminus\let\`{\@tabrj\let\'{\@tablab
73   \let\\=\@tabcr
74   \chightab\firstab
75   \global\@nxttabmar\firstab
76   \dimen\firstab\@totalleftmargin
77   \global\@tabpush\z@\global\@rjfieldfalse
78   \trivlist \item\relax
79   \if@minipage\else\vskip\parskip\fi
80   \setbox\@tabfbox\hbox{%
81     \rlap{\hskip\@totalleftmargin\indent\the\everypar}}%
82   \def\@itemfudge{\box\@tabfbox}%
83   \startline\ignorespaces}

\endtabbing 84 \def\endtabbing{%
85   \stopline\ifnum\@tabpush >\z@\badpoptabs \fi\endtrivlist}

Omitted \global added to \@rtab 17 Jun 86

\@rtab 86 \def\@rtab{\@stopfield\@addfield\ifnum \@curtab<\chightab
87   \global\advance\@curtab \one\else\badtab\fi
88   \tempdima\dimen\@curtab
89   \advance\@tempdima -\dimen\@curtabmar
90   \advance\@tempdima -\wd\curline
91   \global\setbox\curline\hbox{\unhbox\curline\hskip\@tempdima}%
92   \startfield\ignorespaces}

\@settab 93 \def\@settab{\@stopfield\@addfield
94   \ifnum \@curtab <\maxtab
95   \ifnum\@curtab =\chightab
96     \advance\chightab \one
97   \fi
98   \global\advance\@curtab \one
99   \else
100    \latexerror{Tab overflow}\ehd
101   \fi
102   \dimen\@curtab \dimen\@curtabmar
103   \advance\dimen\@curtab \wd\curline
104   \startfield
105   \ignorespaces}

\@ltab 106 \def\@ltab{\@ifatmargin\ifnum\@curtabmar >\@firstab
107   \global\advance\@curtab \m@ne \global\advance\@curtabmar \m@ne\else
108   \badtab\fi\else
109   \latexerror{\string\<\space in mid line}\ehd\fi\ignorespaces}

\@tabplus 110 \def\@tabplus{%
111   \ifnum\@nxttabmar<\chightab

```

```

112      \global\advance\@nxttabmar\@ne
113  \else
114      \@badtab
115  \fi
116  \ignorespaces}

\@tabminus 117 \def\@tabminus{%
118   \ifnum\@nxttabmar>\@firsttab
119     \global\advance\@nxttabmar\m@ne
120   \else
121     \@badtab
122   \fi
123   \ignorespaces}

\@tabrj 124 \def\@tabrj{%
125   \@stopfield\@addfield\global\@rjfieldtrue\@startfield\ignorespaces}

\setbox\@curline made \global in \@tablab. 17 Jun 86

\@tablab 126 \def\@tablab{%
127   \@stopfield
128   \global\setbox\@curline\hbox{%
129     \box\@curline
130     \hskip-\wd\@curfield \hskip-\tabbingsep
131     \box\@curfield
132     \hskip\tabbingsep}%
133   \@startfield
134   \ignorespaces}

135 </2ekernel>
136 <*2ekernel | latexrelease>
137 <latexrelease>\IncludeInRelease{2019/10/01}%
138 <latexrelease>          {\pushtabs}{Make commands robust}%

\pushtabs 139 \DeclareRobustCommand\pushtabs{%
140   \@stopfield\@addfield\global\advance\@tabpush \@ne \begingroup
141   \@contfield}

```

It is, in some sense, an error if, after the endgroup, the current tab setting is higher than the new value of \chightab (which is a local variable). That this is allowed is a fundamental design flaw which is not going to be corrected now.

```

142 \DeclareRobustCommand\poptabs{\@stopfield\@addfield
143   \ifnum \@tabpush >\z@
144     \endgroup
145     \global\advance\@tabpush \m@ne
146     \ifnum \@curtab >\chightab
147       \global \@curtab \chightab
148       \@badtab
149     \fi
150   \else
151     \@badpoptabs
152   \fi
153   \@contfield}

```

```

154 \DeclareRobustCommand\kill{\@stopfield\@startline\ignorespaces}

(End of 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 of definition for \tabbingsep.)

```

1.2 array and tabular environments

Historical L^AT_EX 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 placing a strut in every row of height and depth \arraystretch times the height and depth of the strut 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. \arraycolsep or \tabcolsep spacing is suppressed.
*{N}{PRE}	: equivalent to writing N copies of PRE in the preamble. 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^AT_EX 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 of definition for \extracolsep.*)

\array(\array)

168 \def\array{\let\@acol\@arrayacol \let\@classz\@arrayclassz
169 \let\@classiv\@arrayclassiv
170 \let\\@\@arraycr\let\@haligno\@empty\@tabarray}

(*End of 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 of definition for \endarray, \endtabular, and \endtabular*.*)

\tabular(\tabular)

174 \def\tabular{\let\@haligno\@empty\@tabular}

(*End of definition for \tabular.*)

\tabular*

- Note that the change to use \setlength slightly alters the timing of the expansion and use of the length in #1 but this is very unlikely to have any practical effect.

175 \namedef{tabular*}{#1}{%
176 \setlength\dimen@{#1}{%
177 \edef\@haligno{to\the\dimen@}\@tabular}

(*End of 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 of definition for \@tabular.*)

\@tabarray RmS 91/11/04 added \m@th.

181 \def\@tabarray{\m@th\@ifnextchar[\@array{\@array[c]}}

(*End of 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 of definition for `\@array`.)

`\@arraycr` Array version of `\``.

```
201 \protected\def\@arraycr{%
202   ${}\ifnum0='}\fi\@ifstar\@xarraycr\@xarraycr}
```

(End of definition for `\@arraycr`.)

`\@arraycr`

```
203 \def\@xarraycr{\@ifnextchar[\@garraycr{\ifnum0='}\fi$}\cr}}
```

(End of definition for `\@arraycr`.)

`\@garraycr`

```
204 \def\@garraycr[#1]{%
205   \ifnum0='}\fi$}\ifdim #1>\z@ \@xarraycr[#1]\else
206   \@yarraycr[#1]\fi}
```

(End of definition for `\@garraycr`.)

```

\tabularnewline Tabular version of \\.
207 \let\tabularnewline\relax
(End of definition for \tabularnewline.)
```

```

\@tabularcr
208 \protected\def\@tabularcr{%
209   {\ifnum0='}\fi\@ifstar\@xtabularcr\@xtabularcr}
(End of definition for \@tabularcr.)
```

```

\@xtabularcr
210 \def\@xtabularcr{\@ifnextchar[\@argtabularcr{\ifnum0='{\fi}\cr}}
(End of 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 of definition for \@argtabularcr.)
```

```

\@xargarraycr
218 \def\@xargarraycr#1{\@tempdima #1\advance\@tempdima \dp \arstrutbox
219   \vrule \height\z@ \depth\@tempdima \width\z@ \cr}
(End of 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 of definition for \@yargarraycr.)
```

\multicolumn *Historical L^AT_EX 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^AT_EX 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 of definition for \multicolumn.)

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

Codes for classes and character numbers of array, tabular and multicolumn arguments.

Character	Class	Number
c	0	0
l	0	1
r	0	2
	1	-
@	2	-
p	3	-
{@-exp}	4	-
{p-arg}	5	-

\@testpach \foo : expands \foo, which should be an array parameter

token, and sets \chclass and \chnum to its class and number. Uses \lastchclass to distinguish 4 and 5

Preamble error codes

- 0: 'illegal character'
- 1: 'Missing @-exp'
- 2: 'Missing p-arg'

```
\@addamp ==
BEGIN if @firststamp = true then @firststamp := false
else & fi
END

\@mkpream TOKENLIST ==
BEGIN
  @firststamp := T
  \lastchclass := 6
  \@preamble == null
  \@sharp == \relax
  \@protect == BEGIN \noexpand\protect\noexpand END
  \@startpbox == \relax
  \@endpbox == \relax
  \@expast{TOKENLIST}
  for \@nextchar := expand(\reserved@a)
    do \@testpach{\@nextchar}
      case of \chclass
        0 -> \@classz
        1 -> \@classi
        ...
        5 -> \@classv
      end case
      \lastchclass := \chclass
    od
    case of \lastchclass
      0 -> \hskip \arraycolsep % lrc
      1 -> % |
      2 -> \@preamerr1 % 'Missing @-exp' % @
      3 -> \@preamerr2 % 'Missing p-arg' % p
      4 -> % @-exp
      5 -> \hskip \arraycolsep % p-exp
    end case
  END

\@arrayclassz ==
BEGIN
  \@preamble := \@preamble *
  case of \lastchclass
    0 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
    1 -> \@addamp \hskip \arraycolsep
    2 -> % impossible
```

```

            3 -> % impossible
            4 -> \@addamp
            5 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
            6 -> \@addamp \hskip \arraycolsep
        end case
    * case of \@chnum
        0 -> \hfil$\relax\sharp$\hfil
        1 -> $\relax\sharp$\hfil
        2 -> \hfil$\relax\sharp$\hfil
    end case
END

\@tabclassz == similar to \@arrayclassz

\@classi ==
BEGIN
    \@preamble := \@preamble *
    case of \@lastchclass
        0 -> \hskip \arraycolsep \arrayrule
        1 -> \hskip \doublerulesep \arrayrule
        2 -> % impossible
        3 -> % impossible
        4 -> \arrayrule
        5 -> \hskip \arraycolsep \arrayrule
        6 -> \@arrayrule
    end case
END

\@classii ==
BEGIN
    \@preamble := \@preamble *
    case of \@lastchclass
        0 ->
        1 -> \hskip .5\arrayrulewidth
        2 -> % impossible
        else ->
    end case
END

\@classiii ==
BEGIN
    \@preamble := \@preamble *
    case of \@lastchclass
        0 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
        1 -> \@addamp \hskip \arraycolsep
        2 -> % impossible
        3 -> % impossible
        4 -> \@addamp
        5 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
        6 -> \@addamp \hskip \arraycolsep

```

```

        end case
END

\@arrayclassiv ==
BEGIN \@preamble := \@preamble * $ \@nextchar$ END

\@tabclassiv == same as \@arrayclassv except without the $ ... $

\@classv ==
BEGIN
  \@preamble :=
    \@preamble * \@startpbox{\@nextchar}\ignorespaces\@sharp
      \@endpbox
END

\@expast{S}:
Sets \reserved@a := S with all instances of *{N}{STRING}
replaced by N copies of STRING, where N > 0. An *
appearing inside braces is ignored, but *-expressions
inside STRING are expanded, so nested *-expressions are
handled properly.

\@expast{S} == BEGIN \@xexpast S *0x\@c END

\@xexpast S1 *{N}{S2} S3 \@c ==
BEGIN
  \reserved@a := S1
  \tempcnta := N
  if \tempcnta > 0
    then while \tempcnta > 0 do \reserved@a := \reserved@a S2
        \tempcnta := \tempcnta - 1 od
    \reserved@b == \@xexpast
  else \reserved@b == \@xexnoop
  fi
  \expandafter \reserved@b \reserved@a S3 \@c
END
End of historical LATEX 2.09 comments.
```

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

(End of definition for \@xexnoop.)

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

(End of 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 of 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 of 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 of 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 of 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\$fi\}}

```

(End of 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 of 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 of definition for \@classi.)
```

\@classii

```

321 \def\@classii{%
322   \ifcase\@lastchclass
323     \or
324       \addtopreamble{\hspace{.5\arrayrulewidth}%
325     \fi}

(End of 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 of definition for \@classiii.)
```

\@tabclassiv

```

330 \def\@tabclassiv{\addtopreamble\@nextchar}

(End of definition for \@tabclassiv.)
```

\@arrayclassiv

```

331 \def\@arrayclassiv{\addtopreamble{$\@nextchar$} }
```

(End of definition for \@arrayclassiv.)

\@classv

```

332 \def\@classv{\addtopreamble{\startpbox{\@nextchar}\ignorespaces
333 \sharp\endpbox}}
```

(End of definition for \@classv.)

\@addtopreamble

```

334 \def\@addtopreamble#1{\edef\@preamble{\@preamble #1}}
```

(End of definition for \@addtopreamble.)

\@chclass
\@lastchclass
\@chnum

```

335 \newcount\@chclass
336 \newcount\@lastchclass
337 \newcount\@chnum
```

(End of definition for \@chclass, \@lastchclass, and \@chnum.)

\arraycolsep
\tabcolsep
\arrayrulewidth
\doublerulesep

```

338 \newdimen\arraycolsep
339 \newdimen\tabcolsep
340 \newdimen\arrayrulewidth
341 \newdimen\doublerulesep
```

(End of definition for \arraycolsep and others.)

```
\arraystretch  
342 \def\arraystretch{1} % Default value.
```

(End of definition for \arraystretch.)

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

(End of definition for \@arstrutbox and \@arstrut.)

```
\@arrayrule  
346 \def\@arrayrule{\@addtopreamble{\hskip -.5\arrayrulewidth  
347   \vrule \width \arrayrulewidth\hskip -.5\arrayrulewidth}}
```

(End of 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 of definition for \@testpatch.)

```
\hline  
358 \def\hline{  
359   \noalign{\ifnum0='}\fi\hrule \height \arrayrulewidth \futurelet  
360   \reserved@a\@xhline}
```

(End of 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 of definition for \@xhline.)

```
\vline  
366 \def\vline{\vrule \width \arrayrulewidth}
```

(End of definition for \vline.)

`\cline` The old L^AT_EX2.09 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 of 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 of definition for `\mscount`.)

`\multispan` Modify `\multispan` slightly from its plain T_EX 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 of 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 of definition for \@startpbox and \@endpbox.)

```
\@@startpbox
\@@endpbox
386 \let\@@startpbox=\@startpbox
387 \let\@@endpbox=\@endpbox
```

(End of definition for \@startpbox and \@endpbox.)

388 ⟨/2ekernel⟩

File M

ltpictur.dtx

1 Picture Mode

Picture mode commands. In addition to the commands available in L^AT_EX2.09, This section adds the new \qbezier command for drawing curves.

\qbezier \qbezier[$\langle N \rangle$] ($\langle AX,AY \rangle$) ($\langle BX,BY \rangle$) ($\langle CX,CY \rangle$) plots a quadratic Bezier curve from ($\langle AX,AY \rangle$) to ($\langle CX,CY \rangle$), with ($\langle BX,BY \rangle$) as the third Bezier point, using $N+1$ points equally spaced parametrically. If $N = 0$ (the default value), then a sufficient number of points are used to draw a connected curve—except that at most \qbeziermax+1 points are drawn. A “point” is a square of side \@wholewidth.

\bezier In addition, to be compatible with the old **bezier** package, a variant of this command, \bezier, is defined, in which the first argument is not optional.

Historical L^AT_EX 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^AT_EX 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 of definition for \@wholewidth and \@halfwidth.)

```

\unitlength
5 \newdimen\unitlength \unitlength =1pt

```

(End of definition for \unitlength.)

```

\@picbox
\@picht
6 \newbox\@picbox
7 \newdimen\@picht

```

(End of definition for \@picbox and \@picht.)

\@defaultunitsset Set a length register, #1, accepting number or an etex length expression, #2, with default unit, #3.

The register name in #1 can be prefixed by \advance so that the register is incremented by the supplied value.

```
\@defaultunitsset{\advance\@vxx}{\textwidth-15pt}\unitlength  
#3 can be a literal unit such as cm or a length register such as \unitlength.
```

This is used in all `picture` commands that take picture coordinates. So `\put(2,2)` as previously but now `\put(\textwidth-5cm,0.4\textheight)` Note that you can only use expressions with lengths, `\put(1+2,0)` is not supported.

```
8 </2ekernel>  
9 <*2ekernel | latexrelease>  
10 <latexrelease>\IncludeInRelease{2020/10/01}%  
11 <latexrelease> {\@defaultunitsset{default units} %  
12 \def\@defaultunitsset#1#2#3{ %  
13 \@defaultunits#1\dimexpr#2#3\relax\relax\@nnil}  
14 </2ekernel | latexrelease>  
15 <latexrelease>\EndIncludeInRelease  
16 <latexrelease>\IncludeInRelease{0000/00/00}%  
17 <latexrelease> {\@defaultunitsset{default units} %  
18 <latexrelease>\let\@defaultunitsset\@undefined  
19 <latexrelease>\EndIncludeInRelease  
20 <*2ekernel>
```

(End of definition for `\@defaultunitsset`.)

`pict\picture`) #1 should be white space.

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

```
\pictur@ 21 \long\def\picture#1{\pictur@#1}  
22 \def\pictur@(#1){%  
23 \@ifnextchar({\@picture(#1)}{\@picture(#1)(0,0)})
```

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

```

38  \end{macro}
39  \end{macro}
40  \def\@picture#1#2#3#4{%
41    \picht#2\unitlength
42    \setbox\@picbox\hb@xt@#1\unitlength\bgroup
43    \hskip -#3\unitlength
44    \lower #4\unitlength\hbox\bgroup
45    \ignorespaces}
46  \end{macro}
47  \end{macro}
48  \end{macro}

```

(End of definition for `\@picture`.)

`\endpicture`

```

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

```

(End of 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{%
82    \expandafter\let\csname multiput \endcsname\undefined
83    \def\multiput(#1,#2){%
84      \ifdim#1=0pt \def\unitlength{\unitlength}%
85      \ifdim#2=0pt \def\unitlength{\unitlength}%
86      \multiput{}%
87    }%
```

(End of definition for `\multiput`.)

`\@multiput`

```

98  \def\@multiput(#1,#2){%
99    \killglue\multicnt #3\relax
100   \whilenum \multicnt >\z@\do{%
101     \raise\ydim\hb@xt@.0{\kern\@xdim #4\hss}%
102     \advance\multicnt\m@ne
103     \ifnum\multicnt=0\relax
104       \ignorespaces
105     \else
106       \raise\ydim\hb@xt@.0{\kern\@xdim #1\unitlength}%
107       \raise\ydim\hb@xt@.0{\kern\@xdim #2\unitlength}%
108     \fi
109   }%
110 }
```

(End of definition for `\@multiput`.)

`\@killglue`

```

123  \def\@killglue{\unskip\whiledim \lastskip >\z@\do{\unskip}}
```

(End of 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 of definition for \thinlines and \thicklines.)

\linethickness
130 \DeclareRobustCommand*\linethickness[1]
131   {\@wholewidth #1\relax \@halfwidth .5\@wholewidth \ignorespaces}

(End of definition for \linethickness.)

\isshortstack
132 \def\shortstack{\@ifnextchar[\@shortstack{\@shortstack[c]}}
(End of 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 of definition for \@ishortstack.)

\@ishortstack
141 \def\@ishortstack#1{\ialign{\mb@l {##}\unskip\mb@r\cr #1\crcr}\egroup}
(End of definition for \@ishortstack.)

\@stackcr
\@ixstackcr 142 \protected\def\@stackcr{\@ifstar\@ixstackcr\@ixstackcr}
143 \def\@ixstackcr{\@ifnextchar[\@istackcr{\cr\ignorespaces}{}}

(End of 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 of 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^AT_EX 2.09 comments.

\if@negarg

157 \newif\if@negarg

(*End of 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){\@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 }
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){\@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 of definition for \line.)

\@sline

```

184 \def\@sline{%
185   \ifnum\@xarg<\z@ \negargtrue \xarg -\xarg \yyarg -\yarg
186   \else \negargfalse \yyarg \yarg \fi
187   \ifnum \yyarg >\z@ \tempcpta\yyarg \else \tempcpta -\yyarg \fi
188   \ifnum\tempcpta>6 \badlinearg\tempcpta\z@ \fi
189   \ifnum\@xarg>6 \badlinearg\@xarg \ne \fi
190   \setbox\@linechar\hbox{\@linefnt\getlinechar(\@xarg,\@yyarg)}%

```

If we have something like \line(5,5){30} the \@linechar will not contain a char and later on we will end in an infinite loop. So we check the width of the box and put in something as an emergency fix if necessary.

```

191 \ifdim\wd\@linechar=\z@
192   \setbox\@linechar\hbox{.}%
193   \badlinearg
194 \fi
195 \ifnum \yarg >\z@ \let\upordown\raise \clnht\z@
196   \else\let\upordown\lower \clnht \ht\@linechar\fi
197 \clnwd \wd\@linechar
198 \if@negarg
199   \hskip -\wd\@linechar \def\reserved@a{\hskip -2\wd\@linechar}%
200 \else
201   \let\reserved@a\relax
202 \fi
203 \whiledim \clnwd <\linelen \do
204   {\upordown\clnht\copy\@linechar
205   \reserved@a
206   \advance\clnht \ht\@linechar
207   \advance\clnwd \wd\@linechar}%
208 \advance\clnht -\ht\@linechar
209 \advance\clnwd -\wd\@linechar
210 \tempdima\linelen\advance\tempdima -\clnwd
211 \tempdimb\tempdima\advance\tempdimb -\wd\@linechar
212 \if@negarg \hskip -\tempdimb \else \hskip \tempdimb \fi
213 \multiply\tempdima \m
214 \tempcpta\tempdima
215 \tempdima \wd\@linechar \divide\tempcpta \tempdima
216 \tempdima \ht\@linechar \multiply\tempdima \tempcpta
217 \divide\tempdima \m
218 \advance\clnht \tempdima
219 \ifdim \linelen <\wd\@linechar
220   \hskip \wd\@linechar

```

Warn if line gets so short that it can't be printed. But don't warn if it is exactly zero since that was probably deliberate (e.g., to get a vector head only).

```

221 \ifdim \linelen = \z@
222 \else
223   \picture@warn
224 \fi

```

```

225     \else\@upordown\@clnht\copy\@linechar\fi}

(End of 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 of 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 of 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 of 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 of definition for \@hvector.)
```

```

\@vvector
268 \def\@vvector{\ifnum \oyarg < z0 \@downvector \else \@upvector \fi}
(End of 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,\oyarg)\else \@getrarrow(\oxarg,\oyarg)\fi}%
275 \else\badlinearg\fi}
(End of 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 of 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 of definition for \@getrarrow.)
```

```

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

(End of definition for \@vline.)

\@upline

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

(End of definition for \@upline.)

\@downline

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

(End of definition for \@downline.)

\@upvector

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

(End of definition for \@upvector.)

\@downvector

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

(End of definition for \@downvector.)

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

```
\dashbox{D}{X,Y} ==
BEGIN
leave vertical mode
\hb@xt@ 0pt {
  \baselineskip := 0pt
  \lineskip := 0pt
%% HORIZONTAL DASHES
  \dashdim := X * \unitlength
  \dashcnt := \dashdim + 200 % to prevent roundoff error
  \dashdim := D * \unitlength
  \dashcnt := \dashcnt / \dashdim
  if \dashcnt is odd
    then \dashdim := 0pt
    \dashcnt := (\dashcnt + 1) / 2
  else \dashdim := \dashdim / 2
    \dashcnt := \dashcnt / 2 - 1
  \box\@dashbox := \hbox{\vrule height \@halfwidth
    depth \@halfwidth width \@dashdim}
  \put(0,0){\copy\@dashbox}
  \put(0,Y){\copy\@dashbox}
  \put(X,0){\hskip -\dashdim\copy\@dashbox}
  \put(X,Y){\hskip -\dashdim\box\@dashbox}
  \dashdim := 3 * \dashdim
fi
```

```

\box\@dashbox := \hbox{\vrule height \@halfwidth
                      depth \@halfwidth width D * \unitlength
                      \hskip D * \unitlength}

\@tempcnta := 0
\put(0,0){\hskip \@dashdim
           while \@tempcnta < \@dascnt
             do \copy\@dashbox
                 \@tempcnta := \@tempcnta + 1
             od
         }
\@tempcnta := 0
\put(0,Y){\hskip \@dashdim
           while \@tempcnta < \@dascnt
             do \copy\@dashbox
                 \@tempcnta := \@tempcnta + 1
             od
         }

%% vertical dashes
\@dashdim := Y * \unitlength
\@dashcnt := \@dashdim + 200 % to prevent roundoff error
\@dashdim := D * \unitlength
\@dashcnt := \@dashcnt / \@dashdim
if \@dashcnt is odd
  then \@dashdim := 0pt
      \@dashcnt := (\@dashcnt + 1) / 2
  else \@dashdim := \@dashdim / 2
      \@dashcnt := \@dashcnt / 2 - 1
\box\@dashbox := \hbox{\hskip -\@halfwidth
                      \vrule width \@wholewidth
                      height \@dashdim }

\put(0,0){\copy\@dashbox}
\put(X,0){\copy\@dashbox}
\put(0,Y){\lower\@dashdim\copy\@dashbox}
\put(X,Y){\lower\@dashdim\copy\@dashbox}
\@dashdim := 3 * \@dashdim
fi
\box\@dashbox := \hbox{\vrule width \@wholewidth
                      height D * \unitlength } }

\@tempcnta := 0
\put(0,0){\hskip -\halfwidth
           \vbox{while \@tempcnta < \@dashcnt
                 do \vskip D*\unitlength
                     \copy\@dashbox
                     \@tempcnta := \@tempcnta + 1
                 od
                 \vskip \@dashdim
             } }

\@tempcnta := 0
\put(X,0){\hskip -\halfwidth

```

```

        \vbox{while \tempcnta < \dashcnt
            do \vskip D*\unitlength
                \copy\dashbox
                \tempcnta := \tempcnta + 1
            od
            \vskip \dashdim
        }
    }
}      % END DASHES

\@imakepicbox(X,Y)
END
End of historical LATEX 2.09 comments.

```

```

\Dashbox
309  {/2ekernel}
310  {*2ekernel | latexrelease}
311  {latexrelease}\IncludeInRelease{2020/10/01}%
312  {latexrelease}          {\dashbox}{default units}%
313  {latexrelease}\expandafter\let\csname dashbox \endcsname\@undefined
314  \def\dashbox#1(#2,#3){\leavevmode\hb@xt@z@\baselineskip \z@skip
315  \lineskip \z@skip
316  \@defaultunitsset\@dashdim{#2}\unitlength
317  \dashcnt \@dashdim \advance\@dashcnt 200
318  \@defaultunitsset\@dashdim{#1}\unitlength
319  \divide\@dashcnt \@dashdim
320  \ifodd\@dashcnt\@dashdim \z@
321  \advance\@dashcnt \one \divide\@dashcnt \tw@
322  \else \divide\@dashdim \tw@ \divide\@dashcnt \tw@
323  \advance\@dashcnt \m@ne
324  \setbox\dashbox \hbox{\vrule \height \halfwidth \depth \halfwidth
325  \width \@dashdim}\put(0,0){\copy\dashbox}%
326  \put(0,#3){\copy\dashbox}%
327  \put(#2,0){\hskip-\@dashdim\copy\dashbox}%
328  \put(#2,#3){\hskip-\@dashdim\box\dashbox}%
329  \multiply\@dashdim \thr@@
330  \fi
331  \setbox\dashbox \hbox{%
332  \@defaultunitsset\@tempdimc{#1}\unitlength
333  \vrule \height \halfwidth \depth \halfwidth \width \@tempdimc
334  \hskip\@tempdimc}%
335  \tempcnta\z@
336  \put(0,0){\hskip\@dashdim \whilenum \tempcnta <\@dashcnt
337  \do{\copy\dashbox\advance\tempcnta \one } }\@tempcnta\z@
338  \put(0,#3){\hskip\@dashdim \whilenum \tempcnta <\@dashcnt
339  \do{\copy\dashbox\advance\tempcnta \one } }%
340  \@defaultunitsset\@dashdim{#3}\unitlength
341  \dashcnt \@dashdim \advance\@dashcnt 200
342  \@defaultunitsset\@dashdim{#1}\unitlength
343  \divide\@dashcnt \@dashdim
344  \ifodd\@dashcnt\@dashdim \z@
345  \advance\@dashcnt \one \divide\@dashcnt \tw@
346  \else

```

```

347 \divide\@dashdim \tw@ \divide\@dashcnt \tw@
348 \advance\@dashcnt \m@ne
349 \setbox\@dashbox\hbox{\hskip -\@halfwidth
350 \vrule \@width \@wholewidth
351 \height \@dashdim}\put(0,0){\copy\@dashbox}%
352 \put(#2,0){\copy\@dashbox}%
353 \put(0,#3){\lower\@dashdim\copy\@dashbox}%
354 \put(#2,#3){\lower\@dashdim\copy\@dashbox}%
355 \multiply\@dashdim \thr@@
356 \fi
357 \defaultunitsset@\tempdimb{#1}\unitlength
358 \setbox\@dashbox\hbox{%
359   \vrule \@width \@wholewidth \height\@tempdimb}%
360 \tempcpta\z@
361 \put(0,0){\hskip -\@halfwidth \vbox{\@whilenum \tempcpta <\@dashcnt
362 \do{\vskip\@tempdimb\copy\@dashbox\advance\tempcpta \z@ }%
363 \vskip\@dashdim}}\tempcpta\z@
364 \put(#2,0){\hskip -\@halfwidth \vbox{\@whilenum \tempcpta<\@dashcnt
365 \do{\vskip\@tempdimb\copy\@dashbox\advance\tempcpta \z@ }%
366 \vskip\@dashdim}}\makepicbox(#2,#3)}
367 </2ekernel | latexrelease>

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

```

401 〈\latexrelease〉\advance\@dashcnt \@ne \divide\@dashcnt \tw@
402 〈\latexrelease〉\else
403 〈\latexrelease〉\divide\@dashdim \tw@ \divide\@dashcnt \tw@
404 〈\latexrelease〉\advance\@dashcnt \m@ne
405 〈\latexrelease〉\setbox\@dashbox\hbox{\hskip -\@halfwidth
406 〈\latexrelease〉\vrule \@width \@wholewidth
407 〈\latexrelease〉\@height \@dashdim\put(0,0){\copy\@dashbox}%
408 〈\latexrelease〉\put(#2,0){\copy\@dashbox}%
409 〈\latexrelease〉\put(0,#3){\lower\@dashdim\copy\@dashbox}%
410 〈\latexrelease〉\put(#2,#3){\lower\@dashdim\copy\@dashbox}%
411 〈\latexrelease〉\multiply\@dashdim \thr@@
412 〈\latexrelease〉\fi
413 〈\latexrelease〉\setbox\@dashbox\hbox{\vrule \@width \@wholewidth
414 〈\latexrelease〉\@height #1\unitlength}\@tempcnta\z@
415 〈\latexrelease〉\put(0,0){%
416 〈\latexrelease〉 \hskip -\@halfwidth \vbox{\@whilenum \@tempcnta <\@dashcnt
417 〈\latexrelease〉 \do{\vskip #1\unitlength\copy\@dashbox
418 〈\latexrelease〉 \advance\@tempcnta\@ne }%
419 〈\latexrelease〉 \vskip\@dashdim}\@tempcnta\z@
420 〈\latexrelease〉\put(#2,0){%
421 〈\latexrelease〉 \hskip -\@halfwidth \vbox{\@whilenum \@tempcnta<\@dashcnt
422 〈\latexrelease〉 \do{\vskip #1\unitlength\copy\@dashbox
423 〈\latexrelease〉 \advance\@tempcnta \@ne }%
424 〈\latexrelease〉 \vskip\@dashdim}}\@makepicbox(#2,#3)
425 〈\latexrelease〉\EndIncludeInRelease
426 〈*2ekernel〉

```

(End of definition for \dashbox.)

Historical L^AT_EX 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
\@covdx   := \ovxx - \@tempdima
\@covdx   := \@covdx/2
\@covdy   := \ovyy - \@tempdima
\@covdy   := \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
}
\@covdx   := \@covdx + \@ovro
\@covdy   := \@covdy + \@ovro
\ht\@tempboxa := \dp\@tempboxa := 0
\@put{-\@covdx}{-\@covdy}{\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^AT_EX 2.09 comments.

\if@ovt If producing the Top Bottom Left or Right of an oval.
\if@ovb 427 \newif\if@ovt
\if@ovl 428 \newif\if@ovb
\if@ovr 429 \newif\if@ovl
430 \newif\if@ovr

(End of 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 of 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 of 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 of definition for \@picture@warn.)

```
\@put  
452 \def\@put#1#2#3{\raise #2\hb@xt@z@{\hskip #1#3\hss}}
```

(End of definition for \@put.)

```
\oval  
453 \def\oval(#1,#2){\@ifnextchar[{\@oval(#1,#2)}{\@oval(#1,#2)[]}}
```

(End of 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 of 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 of 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 of 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 of definition for \@ovhorz.)

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

(End of 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{#1}\fi\endgroup
623 }{/2ekernel | latexrelease}
624 \end{IncludeInRelease}
625 \IncludeInRelease{0000/00/00}%
626 \end{latexrelease} {\@circle}{default units}%
627 \end{latexrelease} \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 of 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 of definition for `\@dot`.)

`\@circ`

```

657 \def\@circ#1#2{\@tempdima #1\relax \advance\@tempdima .5\p@
658   \@tempcnta\@tempdima \@tempdima \p@
659   \divide\@tempcnta\@tempdima
660   \ifnum\@tempcnta >15\relax \@tempcnta 15\relax \fi
661   \ifnum \@tempcnta >\z@ \advance\@tempcnta\m@ne\fi
662   \advance\@tempcnta #2\relax
663   \circ\@tempcnta \char\@tempcnta}

```

(End of definition for `\@circ`.)

`\@xarg` Counters used for manipulating the ‘slope’ arguments.

```

664 \newcount\@xarg
665 \newcount\@yarg
666 \newcount\@yyarg

```

(End of definition for `\@xarg`, `\@yarg`, and `\@yyarg`.)

`\@multicnt` Counter used in `\multiput`, and also `\multicolumn`.

```

667 \newcount\@multicnt

```

(End of definition for `\@multicnt`.)

```

\@xdim Length registers.
\@ydim 668 \newdimen\@xdim
669 \newdimen\@ydim

(End of definition for \@xdim and \@ydim.)
```

\@linechar Box for holding a line segment character, for sloping lines.
670 \newbox\@linechar

(End of definition for \@linechar.)

\@linelen Length of the line currently being built.
671 \newdimen\@linelen

(End of definition for \@linelen.)

\@clnwd Height and width of current line segment.
\@clnht 672 \newdimen\@clnwd
673 \newdimen\@clnht

(End of definition for \@clnwd and \@clnht.)

\@dashdim \dashbox internal registers.
\@dashbox 674 \newdimen\@dashdim
\@dashcnt 675 \newbox\@dashbox
676 \newcount\@dashcnt

(End of 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^AT_EX 2.09 comments (not necessarily accurate any more):

```

\qbezier[N] == \bezier{N}

\bezier{N}(AX,AY)(BX,BY)(CX,CY) ==
BEGIN
  IF N = 0
    THEN \@xima := |BX - AX|
          \@xb := |CX - BX|
          \@xa := Max(\@xa, \@xb)
          \@ya := |BY - AY|
          \@yb := |CY - BY|
          \@ya := Max(\@ya, \@yb)
          @sc := Max(\@xa, \@ya)
    %% The coefficient .5 below is the degree of overlap of
    %% successive points, where 1 is no overlap and 0 is
```

```

%% complete overlap. A coefficient of C multiplies
%% the number of points plotted by 1/C.
%%
\@xa := .5 * \halfwidth
@sc := @sc / \halfwidth
@sc := Max(@sc, qbeziermax)
ELSE @sc := N
@scp := @sc+1
\@xb := 2 * (BX - AX) * \unitlength
\@xa := ((CX-AX)*\unitlength - \@xb)/@sc
\@yb := 2 * (BY - AY) * \unitlength
\@ya := ((CY-AY)*\unitlength - \@yb)/@sc
\@pictdot := square rule of width \wholewidth
\count@ := 0
WHILE \count@ < @scp
DO  \@xdim := ((\count@*\@xa + @xb) / @sc) * \count@
\@ydim := ((\count@*\@ya + @yb) / @sc) * \count@
plot pt with relative coords (\@xdim,\@ydim)
\count@ := \count@+1
OD

```

End of historical L^AT_EX 2.09 comments.

\qbeziermax The maximum number of points to plot.

681 \def\qbeziermax{500}

(End of 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 of 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{@ovy{#5}\unitlength
698           \@defaultunitsset{\advance{@ovy}{-#3}\unitlength
699             \ifdim \@ovy<\z@ \@ovy -\@ovy \fi
700           \@defaultunitsset{@ovdy{#7}\unitlength
701             \@defaultunitsset{\advance{@ovdy}{-#5}\unitlength
702               \ifdim \@ovdy<\z@ \@ovdy -\@ovdy \fi
703               \ifdim \@ovy<\@ovdy \@ovy \@ovdy \fi
704             \@multicnt
705               \ifdim \@ovxx>\@ovy \@ovxx \else \@ovy \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{@ovy{#7}\unitlength
722               \@defaultunitsset{\advance{@ovy}{-#3}\unitlength
723                 \advance{@ovy}{-}\@ovdy \divide{@ovy}{@multicnt}

724   \setbox{@tempboxa}\hbox{%
725     \hspace{-}\@halfwidth
726     \vrule \height\@halfwidth
727       \depth \@halfwidth
728       \width \wholewidth}%
729   \put(#2,#3){%
730     \count@\z@
731     \whilenum{\count@<\@tempcpta}{\do
732       {\@xdim\count@\@ovxx
733         \advance{@xdim}{\@ovdx}
734         \divide{@xdim}{@multicnt}
735         \multiply{@xdim}{\count@}

```

```

736      \@ydim\count@\@ovyy
737          \advance\@ydim\@ovdy
738          \divide\@ydim\@multicnt
739          \multiply\@ydim\count@
740          \raise \@ydim
741              \hb@xt@\z@{\kern\@xdim
742                  \unhcopy\@tempboxa\hss}%
743          \advance\count@\@ne}}}
744 /{2ekernel | latexrelease}

745 \end{IncludeInRelease}
746 \IncludeInRelease{0000/00/00} %
747 \bezier{default units}%
748 \def\bezier#1(#2,#3)(#4,#5)(#6,#7){%
749 \ifnum #1=\z@
750 \@ovxx #4\unitlength
751 \advance\@ovxx -#2\unitlength
752 \ifdim \@ovxx<\z@ \@ovxx -\@ovxx \fi
753 \@ovdx #6\unitlength
754 \advance\@ovdx -#4\unitlength
755 \ifdim \@ovdx<\z@ \@ovdx -\@ovdx \fi
756 \ifdim \@ovxx<\@ovdx \@ovxx \@ovdx \fi
757 \@ovyy #5\unitlength
758 \advance\@ovyy -#3\unitlength
759 \ifdim \@ovyy<\z@ \@ovyy -\@ovyy \fi
760 \@ovdy #7\unitlength
761 \advance\@ovdy -#5\unitlength
762 \ifdim \@ovdy<\z@ \@ovdy -\@ovdy \fi
763 \ifdim \@ovyy<\@ovdy \@ovyy \@ovdy \fi
764 \@multicnt
765 \ifdim \@ovxx>\@ovyy \@ovxx \else \@ovyy \fi
766 \@ovxx .5\@halfwidth \divide\@multicnt\@ovxx
767 \ifnum
768 \qbezier{max}{\@multicnt \@multicnt\qbezier{max}\relax}
769 \fi
770 \else \@multicnt#1\relax \fi
771 \@tempcnta\@multicnt \advance\@tempcnta\@ne
772 \@ovdx #4\unitlength \advance\@ovdx -#2\unitlength
773 \multiply\@ovdx \tw@
774 \@ovxx #6\unitlength \advance\@ovxx -#2\unitlength
775 \advance\@ovxx -\@ovdx \divide\@ovxx\@multicnt
776 \@ovdy #5\unitlength \advance\@ovdy -#3\unitlength
777 \multiply\@ovdy \tw@
778 \@ovyy #7\unitlength \advance\@ovyy -#3\unitlength
779 \advance\@ovyy -\@ovdy \divide\@ovyy\@multicnt
780 \setbox\@tempboxa\hbox{%
781 \hskip -\@halfwidth
782 \vrule \height\@halfwidth
783 \depth \@halfwidth
784 \width \@wholewidth} %
785 \put{#2,#3}{%
786 \count@\z@
787 \whilenum{\count@\l\@tempcnta}\do
788 {\@xdim\count@\@ovxx
789 \advance\@xdim\@ovdx

```

```

790 〈\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 of definition for `\bezier` and `\obezier`.)

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 N

ltthm.dtx

1 Theorem Environments

The user creates his own theorem-like environments with the command

`\newtheorem{<name>}{<text>}[<counter>]` or
`\newtheorem{<name>}[<oldname>]{<text>}`

This defines the environment `<name>` to be just as one would expect a theorem environment to be, except that it prints `<text>` instead of “Theorem”.

If `<oldname>` is given, then environments `<name>` and `<oldname>` use the same counter, so using a `<name>` environment advances the number of the next `<name>` environment, and vice-versa.

If `<counter>` is given, then environment `<name>` is numbered within `<counter>`.

E.g., if `<counter>` = `subsection`, then the first `<name>` in subsection 7.2 is numbered `<text> 7.2.1`.

The way `<name>` environments are numbered can be changed by redefining `\the<name>`. *Historical L^AT_EX 2.09 comments (not necessarily accurate any more):*

DOCUMENT STYLE PARAMETERS

`\@thmcOUNTER` : A command such that

`\edef\theCOUNTER{\@thmcOUNTER}`

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

The default is:

`BEGIN \noexpand\arabic{COUNTER} END`

`\@thmcOUNTERsep` : A separator placed between a theorem number and the number of the counter within which it is numbered.

E.g., to make the third theorem of section 7.2 be numbered 7.2-3, `\@thmcOUNTERsep` should be `\def`'`. Its default is `"`.

`\@begintheorem{NAME}{NUMBER}` : A command that begins a theorem environment for a ‘theorem’ named ‘NAME NUMBER’ – e.g., `\@begintheorem{Lemma}{3.7}` starts Lemma 3.7.

`\@opargbegintheorem{NAME}{NUMBER}{OPARG}` :

A command that begins a theorem environment for a ‘theorem’ named ‘NAME NUMBER’ with optional argument OPARG – e.g., `\@begintheorem{Lemma}{3.7}{Jones}` starts ‘Lemma 3.7 (Jones):’.

`\@endtheorem` : A command that ends a theorem environment.

`\newtheorem{NAME}{TEXT}[COUNTER] ==`

`BEGIN`
if `\NAME` is definable

```

then \@definecounter{NAME}
    if COUNTER present
        then \@newctr{NAME}[COUNTER] fi
            \theNAME == BEGIN \theCOUNTER \@thmcOUNTERsep
                eval\@thmcOUNTER{NAME} END
            else \theNAME == BEGIN eval\@thmcOUNTER{NAME} END
                \NAME == \@thm{NAME}{TEXT}
                \endNAME == \@endtheorem
            else error
        fi
    END

\newtheorem{NAME}[OLDNAME]{TEXT} ==
BEGIN
    if counter OLDNAME nonexistent
        then ERROR
    else
        if \NAME is definable
            then BEGIN
                \theNAME == \theOLDNAME
                \NAME == \@thm{OLDNAME}{TEXT}
                \endNAME == \@endtheorem
            END
        else error
    fi
END

\@thm{NAME}{TEXT} ==
BEGIN
    \refstepcounter{NAME}
    if next char =
        then \@ythm{NAME}{TEXT}
        else \@xthm{NAME}{TEXT}
    fi
END

\@xthm{NAME}{TEXT} ==
BEGIN
    \@begintheorem{TEXT}{\theNAME}
    \ignorespaces
END

\@ythm{NAME}{TEXT}[OPARG] ==
BEGIN
    \@opargbegintheorem{TEXT}{\theNAME}{OPARG}
    \ignorespaces
END

```

End of historical L^AT_EX 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^AT_EX book indicates that \newtheorem may be used anywhere in the document...

```

1  {*2ekernel}
2  \def\newtheorem#1{%
3    \@ifnextchar[\{@othm{#1}\}{\@nthm{#1}}}

```

(End of definition for \newtheorem.)

\@nthm

```

4  \def\@nthm#1#2{%
5    \@ifnextchar[\{@xnthm{#1}{#2}\}{\@ynthm{#1}{#2}}}

```

(End of 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 of 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 of 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 of definition for \@othm.)

\@thm

```

26 \def\@thm#1#2{%
27   \refstepcounter{#1}\%
28   \@ifnextchar[\{@ythm{#1}{#2}\}{\@xthm{#1}{#2}}}

```

(End of 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 of definition for \@xthm and \@ythm.)
```

Default values

```

\@thmcnter
\@thmcntersep 33 \def\@thmcnter#1{\noexpand\arabic{#1}}
34 \def\@thmcntersep{.}

(End of definition for \@thmcnter and \@thmcntersep.)
```

\begin{theorem} Providing theorem defaults.

```

\opargbegintheorem 35 \def\@begintheorem#1#2{\trivlist
36   \item[\hspace*{\labelsep}\bfseries #1\ #2]\itshape}
37 \def\@opargbegintheorem#1#2#3{\trivlist
38   \item[\hspace*{\labelsep}\bfseries #1\ #2\ (#3)]\itshape}
39 \def\@endtheorem{\endtrivlist}
40 \end{ekernel}
```

(End of definition for \begin{theorem}, \opargbegintheorem, and \endtheorem.)

File O

ltsect.dtx

1 Sectioning Commands

This file defines the declarations such as `\author` which are used by `\maketitle`. `\maketitle` itself is defined by each class, not in the L^AT_EX kernel.

The second part of the file defines the generic commands used for defining sectioning commands such as `\chapter`. Again the actual document level commands are defined in the class files, in terms of these commands.

```
1  {*2ekernel}
2  \message{title,}
```

1.1 The Title

`\title` The user defines the title and author by the declarations `\title{<name>}`, `\author{<name>}`.
`\author` Similarly the date is declared with `\date{<date>}`.
`\date` Inside these, the `\thanks{<footnote text>}` command may be used to make acknowledgements, notice of address, etc. in a footnote. If there are multiple authors, they have `\and` to be separated with the `\and` command.
`\maketitle` And finally, the `\maketitle` command produces the actual title, using the information previously saved with the other commands.

```
3  /2ekernel
4  {*2ekernel | latexrelease}
5  <latexrelease>\IncludeInRelease{2019/10/01}%
6  <latexrelease>           {\title}{Make commands robust}%
```

`\title` `\title` for use in `\maketitle`. If not given `\maketitle` will produce an error message.
7 `\DeclareRobustCommand\title[1]{\gdef\@title{\#1}}`

(End of 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 of 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 of definition for `\date`.)

`\thanks`
10 `\DeclareRobustCommand\thanks[1]{\footnotemark}`
11 `\protected\@xdef\@thanks{\@thanks`
12 `\protect\footnotetext[\the\c@footnote]{\#1}}%`
13 }

(End of definition for `\thanks`.)

```

\and
14 \DeclareRobustCommand{\and}{%
15   \end{tabular}%
16   \hskip 1em \oplus .17fil%
17   \begin{tabular}[t]{c}}%      \end{tabular}

(End of 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 of definition for \@title.)

\@author
32 \def\@author{\@latex@warning@no@line{No \noexpand\author given}{}}

(End of definition for \@author.)

\@date
33 \gdef\@date{\today}

(End of definition for \@date.)

\@thanks
34 \let\@thanks\empty

(End of definition for \@thanks.)

35 \message{sectioning,}

```

1.2 Sectioning

```

\@secpenalty
36 \newcount\@secpenalty
37 \@secpenalty = -300

(End of 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 of definition for \if@noskipsec and \c@noskipsectrue.)

\@startsection The \@startsection{\name}{\level}{\indent}{\beforeskip}{\afterskip}{\style}*[\althead]{\heading} command is the mother of all the user level sectioning commands. The part after the *, including the * is optional.

name: e.g., 'subsection'

level: a number, denoting depth of section – e.g., chapter = 0, section = 1, etc.

indent: Indentation of heading from left margin

beforekip: Absolute value = skip to leave above the heading. If negative, then paragraph indent of text following heading is suppressed.

afterskip: if positive, then skip to leave below heading, else negative of skip to leave to right of run-in heading.

style: Commands to set style. Since June 1996 release the *last* command in this argument may be a command such as \MakeUppercase or \fbox that takes an argument. The section heading will be supplied as the argument to this command. So setting #6 to, say, \bfseries\MakeUppercase would produce bold, uppercase headings.

If '*' is missing, then increment the counter. If it is present, then there should be no [\althead] 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 ...
          \par{...}
          \endgroup}
```

Pseudocode for the \@startsection command *Historical L^AT_EX 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 of definition for \@startsection.)

\@sect Pseudocode for the \@sect command *Historical L^AT_EX 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 L^AT_EX 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 \seccntformat 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{\@seccntformat{#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 of definition for \@sect.)

\@xsect Pseudocode for the \@xsect command *Historical L^AT_EX 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^AT_EX 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 of 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 of definition for \@seccntformat.)

Pseudocode for the \@sect command *Historical L^AT_EX 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^AT_EX 2.09 comments.

Pseudocode for the \@afterheading command *Historical L^AT_EX 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^AT_EX 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 of definition for `\@ssect`.)

```

\if@afterindent
\@afterindenttrue
124 \newif\if@afterindent \@afterindenttrue

```

(End of 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 of 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 of definition for `\@hangfrom`.)

```

\c@secnumdepth
\c@tocdepth
140 \newcount\c@secnumdepth
141 \newcount\c@tocdepth

```

(End of 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*{...}
- ¹⁴² \def\secdef{\#1{\#2{\@ifstar{\#2}{\@dblarg{\#1}}}}}

(End of 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 of definition for \sectionmark and others.)

¹⁴⁸ \message{contents,}

1.3 Table of Contents etc.

1.3.1 Convention

\tf@{foo} = file number for output for table foo. The file is opened only if @filesw = true.

1.3.2 Commands

A \l@{type}{\{entry\}}{\{page\}} Macro needs to be defined by document style for making an entry of type *type* in a table of contents, etc. E.g., the document style should define \l@chapter, \l@section, etc.

Note: When the \protect command is used in the *entry* or *text* of one of the commands below, it causes the following control sequence to be written on the file without being expanded. The sequence will be expanded when the table of contents entry is processed.

Surprise: Inside an \addcontentsline or \addtocontents command argument, the commands: \index, \glossary, and \label are no-ops. This could cause a problem if the user puts an \index or \label into one of the commands he writes, or into the optional ‘short version’ argument of a \section or \caption command.

\starttoc The \starttoc{\ext} command is used to define the commands:
\tableofcontents, \listoffigures, etc.

For example: \starttoc{lof} is used in \listoffigures. This command reads the .\ext file and sets up to write the new .\ext file.

Historical L^AT_EX 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^AT_EX 2.09 comments.

```

149 \def\@starttoc#1{%
150   \begingroup
151     \makeatletter
152     \cinput{\jobname.#1}%
153     \if@filesw
154       \expandafter\newwrite\curname tf@#1\endcsname
155       \immediate\openout \curname tf@#1\endcsname \jobname.#1\relax
156     \fi
157     \nobreakfalse
158   \endgroup}

```

(End of 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 of 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 of 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 of 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      \hbox@\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 of 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 of 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 of definition for \numberline.)

File P

ltfloat.dtx

1 Floats

The different types of floats are identified by a *<type>* name, which is the name of the counter for that kind of float. For example, figures are of type ‘figure’ and tables are of type ‘table’. Each *<type>* has associated a positive *<type number>*, which is a power of two. E.g.,

figures might be have type number 1, tables type number 2, programs type number 4, etc.

The locations where a float can go are specified by a *<placement specifier>*, which is a list of the possible locations, each denoted by a letter as follows:

- h : here — at the current location in the text.
- t : top — at the top of a text page.
- b : bottom — at the bottom of a text page.
- p : page — on a separate float page

In addition, in conjunction with these, you can use ‘!’ which means that the current values of the float positioning parameters are ignored for this float. (Has no effect on ‘p’, float page positioning.) For example, ‘pht’ specifies that the float can appear in any of three locations: page, here or top.

1.1 Floating Environments

```
1  {*2ekernel}
2  \message{floats,}
```

Historical L^AT_EX 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.
\cdbltopnumber, \dbltopfraction
                  : Same as above, but for double-column floats.
\c@bottomnumber, \bottomfraction
                  : Same as above for bottom of page.
\c@totalnumber   : Number of floats allowed in a single column,
                  including in-text floats.
{textfraction}   : Minimum fraction of column that must contain text.
{floatpagefraction}: Minimum fraction of page that must be taken
                   up by float page.
{dblfloatpagefraction}
                  : Same as above, for double-column floats.
```

The document style must define the following.

```
\fps@TYPE   : The default placement specifier for floats of type
               TYPE.

\ftype@TYPE : The type number for floats of type TYPE.

\ext@TYPE   : The file extension indicating the file on which the
               contents list for float type TYPE is stored.
               For example, \ext@figure = 'lof'.

\fnum@TYPE  : A macro to generate the figure number for a caption.
               For example, \fnum@TYPE == Figure \thefigure.

\@makecaption{NUM}{TEXT} :
               A macro to make a caption, with NUM the value
               produced by \fnum@... and TEXT the text of the caption.
               It can assume it's in a \parbox of the appropriate width.

\@float{TYPE}[PLACEMENT] : This macro begins a float environment for a
               single-column float of type TYPE with PLACEMENT as the placement
               specifier. The default value of PLACEMENT is defined by
               \fps@TYPE. The environment is ended by \end@float.
               E.g., \figure == \@float{figure}, \endfigure == \end@float.

\@float{TYPE}[PLACEMENT] ==
BEGIN
  if hmode then \@bsphack
    \@floatpenalty := -10002
  else \@floatpenalty := -10003
  fi
  \@capttype ==L TYPE
  \@dblflset
  \@fps ==L PLACEMENT
  \@onellevel@sanitize \@fps
  add default PLACEMENT if at most ! in PLACEMENT ==
\@fpsadddefault
  if inner
    then LaTeX Error: 'Not in outer paragraph mode.'
    \@floatpenalty := 0
  else if \@freelist nonempty
    then \@currbox :=L head of \@freelist
    \@freelist :=G tail of \@freelist
    \count@\currbox :=G 32*\ftype@TYPE +
               bits determined by PLACEMENT
  else \@floatpenalty := 0
    LaTeX Error: 'Too many unprocessed floats'
  fi
  fi
  \@currbox :=G \color@vbox
```

```

\normalcolor
\vbox{
%% 15 Dec 87 -
%% removed \boxmaxdepth :=L 0pt
%% that made box 0 depth because it screwed
%% things up. Instead, added \vskip0pt at end
\hsize = \columnwidth
\@parboxrestore
\@floatboxreset
END

\caption ==
BEGIN
\refstepcounter{@capter}
\@dblarg{@caption{@capter}}
END

```

In following definition, `\par` moved from after `\addcontentsline` to before `\addcontentsline` because the `\write` could cause an extra blank line to be added to the paragraph above the caption. (Change made 12 Jun 87)

```

\@caption{TYPE}[STEXT]{TEXT} ==
BEGIN
\par
\addcontentsline{\ext@TYPE}{TYPE}{\numberline{\theTYPE}{STEXT}}
\begingroup
\@parboxrestore
\@normalsize
\@makecaption{\fnum@TYPE}{TEXT}
\par
\endgroup
END

```

`\@dblfloat{TYPE}[PLACEMENT]` : Macro to begin a float environment for a double-column float of type TYPE with PLACEMENT as the placement specifier. The default value of PLACEMENT is 'tp'. The environment is ended by `\end@dblfloat`.
E.g., `\figure*` == `\@dblfloat{figure}`,
`\endfigure*` == `\end@dblfloat`.

```

\@dblfloat{TYPE}[PLACEMENT] ==
Identical to \@float{TYPE}[PLACEMENT] except \hsize and \linewidth
are set to \textwidth.
End of historical LATEX 2.09 comments.

```

`\@floatpenalty`
³ `\newcount\@floatpenalty`

(*End of 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 of 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^AT_EX constructs test for this flag and do not add vertical space when it is true. The intention is that this emulates T_EX'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 of definition for `\@caption`.)

`\@float`

```

\@dblflset 26 \def\@float#1{%
27   \@ifnextchar[%]
28     {\@xflocat{\#1}}%
29     {\edef\reserved@a{\noexpand\@xfloat{\#1}[\csname fps@\#1\endcsname]}%
30      \reserved@a}

```

(End of definition for `\@float` and `\@dblflset`.)

`\@dblfloat`

```

31 \def\@dblfloat{%
32   \if@twocolumn\let\reserved@a\@dbfllt\else\let\reserved@a\@float\fi
33   \reserved@a

```

(End of definition for \@dblfloat.)

\fps@dbl Note that all double floats have default fps ‘tp’.

(End of definition for \fps@dbl.)

\@setfps This sets the fps, dealing with error conditions by adding the default.

(End of definition for \@setfps.)

\@xfloat The first part of this sets the count register that stores all the information about the type and fps of the float.

We assume here that the default specifiers already contain no active characters.

It may be better to store the defaults as numbers, rather than symbol strings.

```
34  </2ekernel>
35  <latexrelease>\IncludeInRelease{2015/01/01}%
36  <latexrelease>                                {\@xfloat}{Check float options}%
37  <2ekernel | latexrelease>
38  \def\@xfloat #1[#2]{%
39    \@nодокумент
40    \def \@capttype {#1}%
41    \def \@fps {#2}%
42    \@onelевел@sanitize \@fps
43    \def \@reserved@b {!}%
44    \ifx \@reserved@b \@fps
45      \@fpsadddefault
46    \else
47      \ifx \@fps \@empty
48        \@fpsadddefault
49      \fi
50    \fi
51    \@ifhmode
52      \@bsphack
53      \@floatpenalty -\@Mii
54    \else
55      \@floatpenalty-\@Miii
56    \fi
57    \@inner
58      \@parmoderr\@floatpenalty\z@
59  \else
60    \@next@\currbox\@freelist
61    {%
62      \@tempcnta \sixt@n
63      \expandafter \tfor \expandafter \reserved@a
64      \expandafter :\expandafter =\@fps
65      \do
```

Start of changes, use a nested if structure, ending in an error.

```
66    {%
67      \if \@reserved@a h%
68        \@ifodd \@tempcnta
69        \else
70          \advance \@tempcnta \one
71        \fi
```

```

72          \else\if \reserved@a t%
73              \@setfpsbit \tw@%
74          \else\if \reserved@a b%
75              \@setfpsbit 4%
76          \else\if \reserved@a p%
77              \@setfpsbit 8%
78          \else\if \reserved@a !%
79              \ifnum \tempcnta>15
80                  \advance\tempcnta -\sixt@@n\relax
81              \fi
82          \else
83              \@latex@error{Unknown float option `\'\reserved@a'}%
84              {Option `\'\reserved@a' ignored and `p' used.}%
85              \@setfpsbit 8%
86          \fi\fi\fi\fi\fi
87      }%

```

End of changes

```

88      \tempcntb \csname ftype@\capttype \endcsname
89      \multiply \tempcntb \xxxii
90      \advance \tempcnta \tempcntb
91      \global \count\currbox \tempcnta
92      }%
93      \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@\@next\@currbox\@freelist
128   <|latexrelease>      \else
129     <|latexrelease>      \@tempcnta \sixt@@n
130   <|latexrelease>      \expandafter \@tfor \expandafter \reserved@a
131     <|latexrelease>      \expandafter :\expandafter =\@fps
132   <|latexrelease>      \do
133     <|latexrelease>      \%
134   <|latexrelease>      \if \reserved@a h%
135     <|latexrelease>      \ifodd \@tempcnta
136   <|latexrelease>      \else
137     <|latexrelease>      \advance \@tempcnta \one
138   <|latexrelease>      \fi
139   <|latexrelease>      \if \reserved@a t%
140     <|latexrelease>      \@setfpsbit \tw@
141   <|latexrelease>      \fi
142   <|latexrelease>      \if \reserved@a b%
143     <|latexrelease>      \@setfpsbit 4%
144   <|latexrelease>      \fi
145   <|latexrelease>      \if \reserved@a p%
146     <|latexrelease>      \@setfpsbit 8%
147   <|latexrelease>      \fi
148   <|latexrelease>      \if \reserved@a !%
149     <|latexrelease>      \ifnum \@tempcnta>15
150       <|latexrelease>      \advance\@tempcnta -\sixt@@n\relax
151     <|latexrelease>      \fi
152   <|latexrelease>      \fi
153   <|latexrelease>      \fi
154   <|latexrelease>      \fi
155   <|latexrelease>      \fi
156   <|latexrelease>      \}%
157   <|latexrelease>      \@tempcntb \csname ftype@\@capttype \endcsname
158   <|latexrelease>      \multiply \@tempcntb \xxxii
159   <|latexrelease>      \advance \@tempcnta \@tempcntb
160   <|latexrelease>      \global \count\@currbox \@tempcnta
161   <|latexrelease>      \}%
162   <|latexrelease>      \@fltovf
163   <|latexrelease>      \fi
164   <|latexrelease>      \global \setbox\@currbox
165   <|latexrelease>      \color@vbox
166   <|latexrelease>      \normalcolor
167   <|latexrelease>      \vbox \bgroup
168   <|latexrelease>      \hsize\columnwidth
169   <|latexrelease>      \parboxrestore
170   <|latexrelease>      \floatboxreset
171   <|latexrelease>\}%

```

```

172  ⟨latexrelease⟩\EndIncludeInRelease
173  ⟨*2ekernel⟩

(End of 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 of definition for \@floatboxreset.)

```
\@setnobreak
179 \def \@setnobreak{%
180   \if@nobreak
181     \let\outer@nobreak\@nobreaktrue
182   \else
183     \fi
184 }
```

(End of definition for \@setnobreak.)

```
\@setminipage
185 \def \@setminipage{%
186   \ominipagetrue
187   \everypar{\ominipagefalse\everypar{}}
188 }
```

(End of 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 of definition for \end@float.)

\end@dblfloat

```

205 </2ekernel>
206 <|latexrelease|\IncludeInRelease{2015/01/01}%
207 <|latexrelease|           {\end@dblfloat}{float order in 2-column}%
208 <|2ekernel | latexrelease>
209 \def\end@dblfloat{%
210     \if@twocolumn
211         \endfloatbox
212         \ifnum\@floatpenalty <\z@
213             \largefloatcheck

```

Force the depth of two column float boxes.

```
214     \global\dp\@currbox1sp %
```

What follows is essentially \end@float without a starting \endfloatbox.

```

215     \cons\@currlist\@currbox
216     \ifnum\@floatpenalty <-\@Mi
217         \penalty -\@Miv
218         \tempdima\prevdepth
219         \vbox{}%
220         \prevdepth\tempdima
221         \penalty\@floatpenalty
222     \else
223         \vadjust{\penalty -\@Miv \vbox{} \penalty\@floatpenalty}\@EspHack
224     \fi
225
226     \fi
227 \else
228     \end@float
229 \fi
230 }%
231 </2ekernel | latexrelease>
232 <|latexrelease|\EndIncludeInRelease
233 <|latexrelease|\IncludeInRelease{0000/00/00}%
234 <|latexrelease|\def\end@dblfloat{%
235 <|latexrelease|\if@twocolumn
236 <|latexrelease| \endfloatbox
237 <|latexrelease| \ifnum\@floatpenalty <\z@

```

We make sure that we never exceed \textheight, otherwise float will never get typeset (91/03/15 FMI).

```

238 <|latexrelease| \largefloatcheck
239 <|latexrelease| \cons\@dbldeferlist\@currbox
240 <|latexrelease| \fi

```

RmS 92/03/18 changed \c@esphack to \c@Ephack.

```
241 〈\textrlease〉    \ifnum \c@floatpenalty =-\c@Mii \c@Ephack\fi  
242 〈\textrlease〉\else  
243 〈\textrlease〉  \end\c@float  
244 〈\textrlease〉\fi  
245 〈\textrlease〉} %  
246 〈\textrlease〉\EndIncludeInRelease  
247 〈*2ekernel〉
```

(End of definition for \end\c@dblfloat.)

\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\c@endbox  
254 }
```

(End of definition for \c@endfloatbox.)

\outer\nobreak

```
255 \let\outer\nobreak\c@empty
```

(End of 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 of definition for \c@largefloatcheck.)

\c@dbf1t

\c@xdblfloat
264 \def\c@dbf1t#1{\c@ifnextchar[{\c@xdblfloat{#1}}{\c@xdblfloat{#1}[tp]}}
265 \def\c@xdblfloat#1[#2]{%
266 \c@xfloat{#1}[#2]\hsize\textrwidth\linewidth\textrwidth}

(End of definition for \c@dbf1t and \c@xdblfloat.)

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 of 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 
```

/2ekernel | texreleas>

```

293 <texreleas>\EndIncludeInRelease
294 <texreleas>      {\@dblfloatplacement}{float order in 2-column}%
295 <texreleas>\def \@dblfloatplacement {%
296 
```

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 
```

(End of definition for \dblfloatplacement.)

Historical L^AT_EX 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 \cparbottom :=G 0
    @reversemargin :=G true
END

\normalmarginpar == BEGIN \cparbottom :=G 0
    @reversemargin :=G false
END

```

End of historical L^AT_EX 2.09 comments.

```

\marginpar
308 \def\marginpar{%
309     \ifhmode
310         \bphack
311         \floatpenalty -\Mii
312     \else
313         \floatpenalty-\Miii
314     \fi
315     \ifinner
316         \parmoderr
317         \floatpenalty\z@
318     \else
319         \next\@currbox\@freelist{}{%
320             \next\@marbox\@freelist{\global\count\@marbox\m@ne}%
321             {\floatpenalty\z@
322             \f@tovf\def\@currbox{\tempboxa}\def\@marbox{\tempboxa}}%
323     \fi
324     \ifnextchar [\@xmpar\@ympar}

```

(End of definition for \marginpar.)

```

\@xmpar
325 \long\def\xmpar[#1]{%
326     \@savemarbox\@marbox{#1}%
327     \@savemarbox\@currbox{#2}%
328     \@xympar}

```

(End of definition for \xmpar.)

```

\@ympar
329 \long\def\ympar#1{%
330     \@savemarbox\@marbox{#1}%
331     \global\setbox\@currbox\copy\@marbox
332     \@xympar}

```

(End of 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 of 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 %
376 \@setnobreak
377 \@setminipage
378 }

```

(End of definition for `\@marginparreset`.)

`\@xympar`

Setting the box here is done only because the code uses `\end@float`; it will be empty and gets discarded.

```

378 \def \@xympar{%
379   \ifnum\@floatpenalty <\z@\@cons\@currlist\@marbox\fi
380   \setbox\@tempboxa
381   \color@vbox
382   \vbox \bgroup
383   \end@float
384   \ignorespacesfalse
385   \esphack
386 }

```

(End of definition for `\@xympar`.)

`\reversemarginpar` `\normalmarginpar`

```

387 \def\reversemarginpar{\global\@mparbottom\z@\@reversemargintrue}
388 \def\normalmarginpar{\global\@mparbottom\z@\@reversemarginfalse}

```

(End of definition for `\reversemarginpar` and `\normalmarginpar`.)

```
389 \message{footnotes,}
```

1.2 Footnotes

Historical L^AT_EX 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 do a negative skip 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^AT_EX 2.09 comments.

\footins L^AT_EX does use the same insert for footnotes as PLAIN.

390 \newinsert\footins

L^AT_EX 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 of definition for \footins.*)

\footnoterule L^AT_EX keeps PLAIN T_EX'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 of definition for \footnoterule.*)

\thefootnote

396 \Qdefinecounter{footnote}

397 \def\thefootnote{\arabic{footnote}}

(*End of 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{mpfootnote}}

(*End of definition for \thempfootnote.*)

\@makefnmark Default definition.

```

400 \%def\@makefnmark{\hbox{$^{\@thefnmark}\m@th$}}
401 \def\@makefnmark{\hbox{\textsuperscript{\normalfont\@thefnmark}}}

```

(End of 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 of 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{\fontsize\sf@size\sf@size#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{\fontsize\sf@size\z@#1}}}}}
417 (latexrelease)\EndIncludeInRelease
418 (*2ekernel)

```

(End of 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 of definition for \textsubscript.)

```

\@textsubscript
 432  </2ekernel>
 433  {*2ekernel | latexrelease}
 434  <latexrelease>\IncludeInRelease{2020/10/01}%
 435  <latexrelease>          {\@textsubscript}{subscript baseline}%
 436  \def\@textsubscript#1{%
 437    {\m@th\ensuremath{_f\mbox{\scriptsize\sffamily\sf@size\sf@size#1}}}}
 438  </2ekernel | latexrelease>
 439  <latexrelease>\EndIncludeInRelease
 440  <latexrelease>\IncludeInRelease{2015/01/01}%
 441  <latexrelease>          {\@textsubscript}{subscript baseline}%
 442  <latexrelease>
 443  <latexrelease>\def\@textsubscript#1{%
 444    {\m@th\ensuremath{_f\mbox{\scriptsize\sffamily\sf@size\z@#1}}}}
 445  <latexrelease>\EndIncludeInRelease
 446  <latexrelease>\IncludeInRelease{0000/00/00}%
 447  <latexrelease>          {\@textsubscript}{subscript baseline}%
 448  <latexrelease>\let\@textsubscript\undefined
 449  <latexrelease>\EndIncludeInRelease
 450  {*2ekernel}>

(End of definition for \@textsubscript.)
```

```
\footnotesep
 451  \newdimen\footnotesep

(End of definition for \footnotesep.)
```

```
\footnote
 452  \def\footnote{\@ifnextchar[\@xfootnote{\stepcounter\@mpfn
 453    \protected\@xdef\@thefnmark{\thempfn}%
 454    \@footnotemark\@footnotetext}}}

(End of 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 of 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 \OMM
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 \OMM
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 \OMM
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 of definition for \footnotetext.)

\footnotemark

```

518 \def\footnotemark{%
519   \ifnextchar[\@xfootnotemark
520     {\stepcounter{footnote}%
521      \protected@xdef\@thefnmark{\thefootnote}%
522      \@footnotemark}%

```

(End of 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 of 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 of definition for `\@footnotemark`.)

`\footnotetext`

```

535 \def\footnotetext{%
536   \ifnextchar [ \@xfootnotenext
537     {\protected@xdef\@thefnmark{\thempfn}%
538     \@footnotetext}}%

```

(End of 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 of definition for `\@xfootnotenext`.)

`\thempfn`

```

545 \def\@mpfn{footnote}
546 \def\thempfn{\thefootnote}%

```

(End of 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 of definition for \footref.*)

```
566 ⟨/2ekernel⟩
```

File Q

ltidxglo.dtx

1 Index and Glossary Generation

Index and Glossary commands.

```
\makeindex      A preamble command to turn on indexing.  
\makeglossary   A preamble command to turn on making glossary entries.  
    \index        Make an index entry for #1.  
    \glossary     Make a glossary entry for #1.  
Historical LATEX 2.09 comments (not necessarily accurate any more):  
\makeindex ==  
  BEGIN  
    \index ==  BEGIN \@bsphack  
    \begingroup  
      \protect{X} == \string X\space  
      %% added 3 Feb 87 for \index commands  
      %% in \footnotes  
      re-\catcode special characters  
      to 'other'  
      \@wrindex  
  END  
  
\@wrindex{ITEM} ==  
  BEGIN  
    write of {\indexentry{ITEM}{page number}}  
    \endgroup  
    \@esphack  
  END  
  
INITIALIZATION:  
  
\index == BEGIN \@bsphack  
  \begingroup  
    re-\catcode special characters (in case '%' there)  
    \@index  
  END  
  
\@index{ITEM} == BEGIN \endgroup \@esphack END  
  
Changes made 14 Apr 89 to write \glossaryentry's instead of  
\indexentry's on the .glo file.  
End of historical LATEX 2.09 comments.  
1 {*2ekernel}  
2 \message{index,}
```

```

\makeindex

3 \def\makeindex{%
4   \newwrite\@indexfile
5   \immediate\openout\@indexfile=\jobname.idx
6   \def\index{\@bsphack\begingroup
7     \@sanitize
8     \wrindex}\typeout
9   {Writing index file \jobname.idx}%

```

Opening the write channel should be done only once since on some OS multiple opens are forbidden and in any case it is useless. So we turn this into a no-op after use.

```

10  \let\makeindex\empty
11 }
12 \onlypreamble\makeindex

```

(End of definition for \makeindex.)

```

\@wrindex

13 \def\@wrindex#1{%
14   \protected@write\@indexfile{}{%
15     \string\indexentry{#1}{\thepage}}%
16   \endgroup
17   \esphack}

```

(End of definition for \@wrindex.)

```

\index

18 \def\index{\@bsphack\begingroup \@sanitize\@index}

```

(End of definition for \index.)

```

\@index

19 \def\@index#1{\endgroup\esphack}

```

(End of 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 of definition for \makeglossary.)

```
\@wrglossary
30 \def\@wrglossary#1{%
31   \protected@write\@glossaryfile{}{%
32     {\string\glossaryentry{\#1}{\thepage}}%
33   \endgroup
34   \@esphack}
(End of definition for \@wrglossary.)
```

```
\glossary
35 \def\glossary{\@bsphack\begingroup\@sanitize\@index}
(End of definition for \glossary.)
36 </2ekernel>
```

File R

ltbibl.dtx

1 Bibliography Generation

A bibliography is created by the `thebibliography` environment, which generates a title such as “References”, and a list of entries. The BIBTEX program will create a file containing such an environment, which will be read in by the `\bibliography` command. With BIBTEX, the following commands will be used.

`\bibliography{<file1,<file2, ...,<filen>}` : specifies the bibdata files. Writes a `\bibdata` entry on the `.aux` file and tries to read in `mainfile.bbl`.

`\bibliographystyle{<style>}` : Writes a `\bibstyle` entry on the `.aux` file.

The `thebibliography` environment is a list environment. To save the use of an extra counter, it should use `enumiv` as the item counter. Instead of using `\item`, items in the bibliography are produced by the following commands:

`\bibitem{<name>}` : Produces a numbered entry cited as `<name>`.

`\bibitem[<label>]{<name>}` : Produces an entry labeled by `<Label>` and cited by `<name>`.

The former is used for bibliographies with citations like [1], [2], etc.; the latter is used for citations like [Knuth82].

The document class must define the `thebibliography` environment. This environment has a single argument, which is the widest bibliography label— e.g., if the [Knuth67] is the widest entry, then this argument will be Knuth67. The `\thebibliography` command must begin a list environment, which the `\endthebibliography` command ends.

`\cite` Entries are cited by the command `\cite{<name>}`.

`\nocite{<citations>}` puts information on the `.aux` file that causes BIBTEX to include the `{<citations>}` list in the bibliography, but puts nothing in the text.

`\nocite{*}` is special: it tells BIBTEX to put the whole of a collection of references into the bibliography.

1 `(*2ekernel)`
2 `\message{bibliography,}`

Historical L^AT_EX 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^AT_EX 2.09 comments.

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

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

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

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

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

\cite
12 </2ekernel>
13 <*2ekernel | latexrelease>
14 <latexrelease>\IncludeInRelease{2022/06/01}%
15 <latexrelease>           {\cite}{check for blank}%
16 \DeclareRobustCommand\cite{%
17   \@ifnextchar [{\@tempswattrue\@citex@checkblank}{\@tempswafalse\@citex@checkblank[]}}}
```

Due to the way `\@for` as used in `\@citex` behaves an empty argument to `\cite` did not produce any warning for a missing citation. So we now inject a command before calling `\@citex` that does the checking for us. It is not done in `\@citex` directly, because that command is altered by a number of packages/classes and this way it is more likely that the check survives.

```
18 \def\@citex@checkblank[#1]{%
19   \IfBlankTF {#2}{%
20     {\@citex[#1]{\space}}{%
21       {\@citex[#1]{#2}}{%
22     }%
23   }%
24 }
```

```

24  ⟨latexrelease⟩\EndIncludeInRelease
25  ⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
26  ⟨latexrelease⟩                                {\cite}{check for blank}%
27  ⟨latexrelease⟩
28  ⟨latexrelease⟩\DeclareRobustCommand\cite{%
29  ⟨latexrelease⟩  \@ifnextchar [{\@tempswatrue\@citex}{\@tempswafalse\@citex[]}}%
30  ⟨latexrelease⟩\let\@citex@checkblank\@undefined
31  ⟨latexrelease⟩
32  ⟨latexrelease⟩\EndIncludeInRelease
33  {*2ekernel}

```

(End of definition for `\cite`.)

`\@citex` \penalty\@m added to definition of `\@citex` to allow a line break after the ‘,’ in citations like [Jones80,Smith77] (Added 23 Oct 86)
space added after the ‘,’ (21 Nov 87)

```

34  \def\@citex[#1]{\leavevmode
35    \let\@citema\@empty
36    \@cite{\@for\@citemb:=#2\do
37      {\@citema\def\@citema{,\penalty\@m\ }%
38       \edef\@citemb{\expandafter\@firstofone\@citemb\@empty}%
39       \if@filesw\immediate\write\@auxout{\string\citation{\@citemb}}\fi

```

Using `\hbox` instead of `\mbox` is fine because of the `\leavevmode` above. In fact the use of a box around the citation contents is more than questionable in my view (FMi), but within 2e I have to keep that for compatibility reasons as it would probably change too many existing documents. Its main reason is to avoid hyphenation of labels such as [FOOB89] into [FOO- B89] so in certain styles it makes sense; but, for example, in author year citations it becomes more than questionable.

So Chris added yet another hook here, as suggested by, at least, Donald Arseneau. Note that this one is inside the first argument of the `\@cite` hook. This decouples the top-level typesetting of the citation from the details of the other business conducted here. All this really needs a complete rethink to get the right modularity.

```

40    \ifundefined{b@\@citemb}{\hbox{\reset@font\bfseries ?}}%
41    \G@refundefinedtrue
42    \@latex@warning
43      {Citation ‘\@citemb’ on page \thepage \space undefined}%
44      {\@cite@ofmt{\csname b@\@citemb\endcsname}}}\#1}}

```

(End of definition for `\@citex`.)

```

\bibdata
\bibstyle 45 \let\bibdata=\@gobble
46 \let\bibstyle=\@gobble

```

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

```

\bibliography
47 \def\bibliography#1{%
48   \if@filesw
49     \immediate\write\@auxout{\string\bibdata{\zap@space#1 \@empty} }%
50   \fi
51   \@input{\jobname.bbl}

```

(End of definition for \bibliography.)

```
\bibliographystyle
52 \def\bibliographystyle#1{%
53   \ifx\@begindocumenthook\@undefined\else
54     \expandafter\AtBeginDocument
55   \fi
56   {\if@filesw
57     \immediate\write\auxout{\string\bibstyle{#1}}%
58   \fi}}
```

(End of definition for \bibliographystyle.)

\nocite (Added 14 Jun 85)

This puts information on the .aux file that causes BIBTEX to include the citation list in the bibliography, but puts nothing in the text.

RmS 93/08/06: Made loop for \nocite like that for \@citet, to get rid of leading spaces.

```
59 </2ekernel>
60 <*2ekernel | latexrelease>
61 <| latexrelease>\IncludeInRelease{2021/06/01}%
62 <| latexrelease>           {\nocite}{Allow nocite in preamble}%
63 \def\nocite#1{\@bsphack
```

With the implementation designed already in LATEX 2.09 the \nocite command will not work before \begin{document} since it tries to write to the .aux file which is not open before that point. As a result the “reference” will appear on the terminal and nothing else will happen.

[This would be easy to fix, but then a document using the fix will silently fail on an older release of LATEX, missing all citations done with \nocite. Thus we do only generate an error message and leave the fix for a LATEX 2 ε successor.]

Given that we are now a quarter century into using LATEX 2 ε there is no good reason any more do limit ourself to 2.09 considerations. So we now simply delay the \nocite if it is issued in the preamble.

```
64 \ifx\@onlypreamble\document
```

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

```
65 \@for\@citereb:=#1\do{%
66   \edef\@citereb{\expandafter\@firstofone\@citereb}%
67   \if@filesw\immediate\write\auxout{\string\citation{\@citereb}}\fi
68   \@ifundefined{b@\@citereb}{\G@refundefinedtrue
69     \G@warning{Citation ‘\@citereb’ undefined}}{}%
70 }
```

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

```
71 \% \@latex@error{Cannot be used in preamble}\@eha
```

Instead we delay the declaration to the start of the document. We have to use a late hook for this, so that it comes after the .aux file is open for writing and after \@preamblecmds was executed to change the above test. Therefore \AtBeginDocument would still be too early.

```
72 \AddToHook{begindocument/end}[kernel]{\nocite{#1}}%
73 \fi
```

```

74   \@esphack}
75   {/2ekernel | latexrelease}
76   \latexrelease\EndIncludeInRelease
77   \latexrelease\IncludeInRelease{0000/00/00}%
78   \latexrelease{}{\nocite}{Allow nocite in preamble}%
79   \latexrelease
80   \latexrelease\def\nocite#1{\@bsphack
81   \latexrelease\ifx\onlypreamble\document
82   \latexrelease\@for\@citeb:=#1\do{%
83   \latexrelease\edef\@citeb{\expandafter\firstofone\@citeb}%
84   \latexrelease\if@filesw\immediate\write\auxout{\string\citation{\@citeb}}\fi
85   \latexrelease\@ifundefined{b@\@citeb}{\G@refundefinedtrue
86   \latexrelease\@latex@warning{Citation ‘\@citeb’ undefined}}{}%
87   \latexrelease\else
88   \latexrelease\@latex@error{Cannot be used in preamble}\@eha
89   \latexrelease\fi
90   \latexrelease\@esphack}
91   \latexrelease
92   \latexrelease\EndIncludeInRelease
93   {*2ekernel}

```

Since `\nocite{*}` should not produce a warning about undefined citation keys (see PR 557), we need to set the control sequence ‘`\b@*`’ to something other than `\relax`. As a result `\cite{*}` will not warn either (but that never worked with BibTeX in the first place).

```
94 \expandafter\let\csname b@\endcsname\empty
```

(*End of definition for `\nocite`.*)

1.1 Default definitions

This hook determines the ‘relative formatting’ of the two logical parts of a citation with comment.

```
\@cite
95 \def\@cite#1#2{[#1\if@tempswa , #2\fi]}%
```

(*End of definition for `\@cite`.*)

`\@cite@ofmt` This is, in general, a command that appears to have one argument whose value is, in the kernel, a single cs whose name is the expansion of `b@\@citeb`; the expansion of this cs will typically be some hmode material that produces the detailed typeset form of just the citations themselves.

```
96 \let\@cite@ofmt\hbox
```

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

```
\@biblabel
```

```
97 \def\@biblabel#1{[#1]}
98 {/2ekernel}
```

(*End of definition for `\@biblabel`.*)

File S

lmarks.dtx

Abstract

Marks are used to communicate information about the content of a page to the output routine. For example, in order to construct running headers, the output routine needs information about which section names are present on a page, and this information is passed to it through the mark system. However, marks may also be used for other purposes. This module provides a generalized mechanism for marks of independent classes.

1 Introduction

The *T_EX* engines offer a low-level mark mechanism to communicate information about the content of the current page to the asynchronous operating output routine. It works by placing `\mark` commands into the source document. When the material for the current page is assembled in box 255, *T_EX* scans for such marks and sets the commands `\topmark`, `\firstmark` and `\botmark`. The `\firstmark` receives the content of the first `\mark` seen in box 255 and `\botmark` the content of the last mark seen. The `\topmark` holds the content of the last mark seen on the previous page or more exactly the value of `\botmark` from the previous page. If there are no marks on the current page then all three are made equal to the `\botmark` from the previous page.

This mechanism works well for simple formats (such as plain *T_EX*) whose output routines are only called to generate pages. It fails, however, in *L^AT_EX* (and other more complex formats), because here the output routine is sometimes called without producing a page, e.g., when encountering a float and placing it into one of the float regions. In that case the output routine is called, determines where to place the float, alters the goal for assembling text material (if the float was added to the top or bottom region) and then it resumes collecting textual material.

As a result the `\botmark` gets updated and so `\topmark` no longer reflects the situation at the top of the next page when that page is finally boxed.

Another problem for *L^AT_EX* was that it wanted to use several “independent” marks and in the early implementations of *T_EX* there was only a single `\mark` command available. For that reason *L^AT_EX* implemented its own mark mechanism where the marks always contained two parts with their own interfaces: `\markboth` and `\markright` to set marks and `\leftmark` and `\rightmark` to retrieve them.

However, this extended mechanism (while supporting scenarios such as chapter/section marks) was far from general. The mark situation at the top of a page (i.e., `\topmark`) remained unusable and the two marks offered were not really independent of each other because `\markboth` (as the name indicates) was always setting both.

The new mechanism overcomes both issues:

- It provides arbitrarily many, fully independent named marks, that can be allocated and, from that point onwards, used.
- It offers access for each such marks to retrieve its top, first, and bottom values separately.
- Furthermore, the mechanism is augmented to give access to marks in different “regions” which may not be just full pages.

2 Design-level and code-level interfaces

The interfaces are mainly meant for package developers, but they are usable (with appropriate care) also in the document preamble, for example, when setting up special running headers with `fancyhdr`, etc. They are therefore available both as CamelCase commands as well as commands for use in the L3 programming layer. Both are described together below.

```
\NewMarkClass \NewMarkClass {<class>}
\mark_new_class:n \mark_new_class:n {<class>}
```

Declares a new `<class>` of marks to be tracked by L^AT_EX. Each `<class>` must be declared before it is used.

Mark classes can only be declared before `\begin{document}`.

```
\InsertMark \InsertMark {<class>} {<text>}
\mark_insert:nn \mark_insert:nn {<class>} {<text>}
```

Adds a mark to the current galley for the `<class>`, containing the `<text>`.

It has no effect in places in which you can't place floats, e.g., a mark inside a box or inside a footnote never shows up anywhere.

If used in vertical mode it obeys L^AT_EX's internal `\nobreak` switch, i.e., it does not introduce a breakpoint if used after a heading. If used in horizontal mode it doesn't handle spacing (like, for example, `\index` or `\label` does, so it should be attached to material that is typeset).

```
insertmark \AddToHook {insertmark} {<code>}
```

When marks are inserted, the mark content may need some special treatment, e.g., by default `\label`, `\index`, and `\glossary` do not expand at this time (but only later if and when the mark content is actually used). In order to allow packages to augment or alter this setup there is a public hook `insertmark` that is executed at this point. It runs in a group so local modification to commands are only applied to the `<text>` argument of `\InsertMark` or `\mark_insert:nn`.

```
\TopMark      * \TopMark  [{<region>}]{<class>}
\FirstMark    * \FirstMark [{<region>}]{<class>}
\LastMark     * \LastMark [{<region>}]{<class>}
\mark_use_top:nn * \mark_use_top:nn  [{<region>}]{<class>}
\mark_use_first:nn * \mark_use_first:nn [{<region>}]{<class>}
\mark_use_last:nn * \mark_use_last:nn [{<region>}]{<class>}
```

These functions expand to the appropriate mark $\langle text \rangle$ for the given $\langle class \rangle$ in the specified $\langle region \rangle$. The default $\langle region \rangle$ in the design-level commands is `page`. Note that with the L3 layer commands there are no optional arguments, i.e., both arguments have to be provided.

TEXhackers note: The result is returned within the `\unexpanded` primitive (`\exp_not:n`), which means that the $\langle text \rangle$ does not expand further when appearing in an `x`-type or `e`-type argument expansion.

The “first” and “last” marks are those seen first and last in the current region/page, respectively. The “top” mark is the last mark of the $\langle class \rangle$ seen in an earlier region, i.e., the $\langle text \rangle$ what would be “current” at the very top of the region.

Important!

The commands are only meaningful inside the output routine, in other places their result is (while not random) unpredictable due to the way L^AT_EX cuts text material into pages.

Currently, $\langle region \rangle$ is one of `page`, `previous-page`, `column`, and `previous-column`. If a page has just been finished then the region `page` refers to the current page and `previous-page`, as the name indicates, to the page that has been finished previously. This means you are able to access mark information for the current page as well as for the page before if you are inside the output routine, without the need to explicitly save that information beforehand.

In single column documents the `column` is the same as the `page` region, but in two-column documents, `column` refers to the current column that just got finished and `previous-column` to the one previously finished. Code for running headers are (in standard L^AT_EX) only evaluated when both columns are assembled, which is another way of saying that in that case `previous-column` refers to the left column and `column` to the right column. However, to make this a bit nicer to access, there are also alias regions named `first-column` and `last-column`³⁶ to access these regions.³⁷

Note that you can only look backwards at already processed regions, e.g., in a `twoside` document finishing a recto (odd, right-hand) page you can access the data from the facing verso (left-hand) page, but if you are finishing a left-hand page you can't integrate data from the upcoming right-hand page. If such a scenario needs to be realized then it is necessary to save the left-hand page temporarily instead of finalizing it, process material for the right-hand page and once both are ready, attach running headers and footers and shipout out both in one go.³⁸

³⁶This is called “last” not “second” in anticipation of extending the mechanism to multiple columns, where first and last would still make sense.

³⁷At the moment there aren't any `previous-...-column` regions to access the columns from the previous page. If necessary, the mechanism could be easily augmented to cover them too, though.

³⁸As of now that scenario is not yet officially supported.

```
\IfMarksEqualTF      * \IfMarksEqualTF  [{<region>} {<class>} {<pos1>} {<pos2>} {<true>} {<false>}]
\mark_if_eq:nnnnTF  * \mark_if_eq:nnnnTF [{<region>} {<class>} {<pos1>} {<pos2>} {<true>} {<false>}]
\mark_if_eq:nnnnnnTF * \mark_if_eq:nnnnnnTF [{<region1>} {<class1>} {<pos1>}
                                              {<region2>} {<class2>} {<pos2>} {<true>} {<false>}]
```

These conditionals allow you to compare the content of two marks and act based on the result. The commands work in an expansion context, if necessary.

It is quite common when programming with marks to need to interrogate conditions such as whether marks have appeared on a previous page, or if there are multiple marks present on the current page, and so on. The tests above allow for the construction of a variety of typical test scenarios, with three examples presented below.

The first two conditionals cover only the common scenarios. Both marks are picked up from the same *<region>* (by default *page*) and they have to be of the same *<class>*.³⁹ The *<pos>* argument can be either *top*, *first*, or *last*.

If you wish to compare marks across different regions or across different classes, you have to do it using the generic test only available in the L3 programming layer or do it manually, i.e., get the marks and then compare the values yourself.⁴⁰

However, the basic version is enough for the following typical use cases:

Test for at most one mark of class *myclass* on current page: If the first and last mark in a region are the same then either there was no mark at all, or there was at most one. To test this on the current page:

```
\NewMarkClass{myclass}
\IfMarksEqualTF{myclass}{first}{last}
  { <zero or one mark> }{ <two or more marks> }
```

Test for no mark of class *myclass* in the previous page: If the top mark is the same as the first mark, there is no mark in the region at all. If we wanted to do this test for the previous page:

```
\IfMarksEqualTF[previous-page]{myclass}{top}{first}
  { <no marks> }{ <at least one mark> }
```

Comparing *top* and *last* would give you the same result.

Test for zero, one, or more than one: Combining the two tests from above you can test for zero, one or more than one mark.

```
\IfMarksEqualTF{myclass}{top}{first}
  { <no marks> }
  {\IfMarksEqualTF{myclass}{first}{last}
    { <exactly one mark> }{ <more than one mark> }}
```

If you need one of such tests more often (or if you want a separate command for it for readability), then consider defining:

```
\providetcommand\IfNoMarkTF[2][page]{\IfMarksEqualTF[#1]{#2}{first}{last}}
```

³⁹If an undeclared mark class is used the tests return *true* (not an error).

⁴⁰If two undeclared mark classes are compared the result is always *true*; if a declared and an undeclared mark class is used it is always *false*.

2.1 Debugging mark code

```
\DebugMarksOn    \DebugMarksOn ...    \DebugMarksOff  
\DebugMarksOff  
\mark_debug_on:  
\mark_debug_off:
```

Commands to turn the debugging of mark code on or off. The debugging output is rather coarse and not really intended for normal use at this point in time.

3 Application examples

If you want to figure out if a break was taken at a specific point, e.g., whether a heading appears at the top of the page, you can do something like this:

```
\newcounter{breakcounter}  
\NewMarkClass{break}  
\newcommand\markedbreak[1]{\stepcounter{breakcounter}%  
    \InsertMark{break}{\arabic{breakcounter}}%  
    \penalty #1\relax  
    \InsertMark{break}{-\arabic{breakcounter}}}
```

To test if the break was taken you can test if `\TopMark{break}` is positive (taken) or negative (not taken) or zero (there was never any marked break so far). The absolute value can be used to keep track of which break it was (with some further coding).

to be extended with additional application examples

4 Legacy L^AT_EX 2 _{ε} interface

Here we describe the interfaces that L^AT_EX 2 _{ε} offered since the early nineties and some minor extensions.

4.1 Legacy design-level and document-level interfaces

```
\markboth \markboth {<left>} {<right>}  
\markright \markright {<right>}
```

L^AT_EX 2 _{ε} uses two marks which aren't fully independent. A "left" mark generated by the first argument of `\markboth` and a "right" mark generated by the second argument of `\markboth` or by the only argument of `\markright`. The command `\markboth` and `\markright` are in turn called from heading commands such as `\chaptermark` or `\sectionmark` and their behavior is controlled by the document class.

For example, in the `article` class with `twoside` in force the `\sectionmark` will issue `\markboth` with an empty second argument and `\subsectionmark` will issue `\markright`. As a result the left mark will contain chapter titles and the right mark subsection titles.

Note, however, that in one-sided documents the standard behavior is that only `\markright` is used, i.e., there will only be right-marks but no left marks!

```
\leftmark * \leftmark
\rightmark * \rightmark
```

These functions return the appropriate mark value from the current page and work as before, that is `\leftmark` will get the last (!) left mark from the page and `\rightmark` the first (!) right mark.

In other words they work reasonably well if you want to show the section title that is current when you are about to turn the page and also show the first subsection title on the current page (or the last from the previous page if there wasn't one). Other combinations can't be shown using this interface.

The commands are fully expandable, because this is how they have been always defined in L^AT_EX. However, this is of course only true if the content of the mark they return is itself expandable and does not contain any fragile material. Given that this can't be guaranteed for arbitrary content, a programmer using them in this way should use `\protected@edef` and *not* `\edef` to avoid bad surprises as far as this is possible, or use the new interfaces (`\TopMark`, `\FirstMark`, and `\LastMark`) which return the *(text)* in `\exp_not:n` to prevent uncontrolled expansion.

4.2 Legacy interface extensions

The new implementation adds three mark classes: `2e-left`, `2e-right` and `2e-right-nonempty` and patches `\markboth` and `\markright` slightly so that they also update these new mark classes, so that the new classes work with existing document classes.

As a result you can use `\LastMark{2e-left}` and `\FirstMark{2e-right}` instead of `\leftmark` and `\rightmark`. But more importantly, you can use any of the other retrieval commands to get a different status value from those marks, e.g., `\LastMark{2e-right}` would return the last subsection on the page (instead of the first as returned by `\rightmark`).

The difference between `2e-right` and `2e-right-nonempty` is that the latter will only be updated if the material for the mark is not empty. Thus `\markboth{title}{}{}` as issued by, say, `\sectionmark`, sets a `2e-left` mark with `title` and a `2e-right` mark with the empty string but does not add a `2e-right-nonempty` mark.

Thus, if you have a section at the start of a page and you would ask for `\FirstMark{2e-right}` you would get an empty string even if there are subsections on that page. But `2e-right-nonempty` would then give you the first or last subsection on that page. Of course, nothing is simple. If there are no subsections it would tell you the last subsection from an earlier page. We therefore need comparison tools, e.g., if top and first are identical you know that the value is bogus, i.e., a suitable implementation would be

```
\IfMarksEqualTF{2e-right-nonempty}{top}{first}
  { <appropriate action if there was no real mark> }
  {\FirstMark{2e-right-nonempty}}
```

5 Notes on the mechanism

In contrast to vanilla T_EX, ε -T_EX extends the mark system to allow multiple independent marks. However, it does not solve the `\topmark` problem which means that L^AT_EX still needs to manage marks almost independently of T_EX. The reason for this is that the more complex output routine used by L^AT_EX to handle floats (and related structures)

means that `\topmark(s)` remain unreliable. Each time the output routine is fired up, `TEX` moves `\botmark` to `\topmark`, and while ε -`TEX` extends this to multiple registers the fundamental concept remains the same. That means that the state of marks needs to be tracked by `LATEX` itself. An early implementation of this package used `TEX`'s `\botmark` only to ensure the correct interaction with the output routine (this was before the ε -`TEX` mechanism was even available). However, other than in a prototype implementation for `LATEX3`, this package was never made public.

The new implementation now uses ε -`TEX`'s marks as they have some advantages, because with them we can leave the mark text within the galley and only extract the marks during the output routine when we are finally shipping out a page or storing away a column for use in the next page. That means we do not have to maintain a global data structure that we have to keep in sync with informational marks in the galley but can rely on everything being in one place and thus manipulations (e.g. reordering of material) will take the marks with them without a need for updating a fragile linkage.

To allow for completely independent marks we use the following procedure:

- For every type of marks we allocate a mark class so that in the output routine `TEX` can calculate for each class the current top, first, and bottom mark independently. For this we use `\newmarks`, i.e., one marks register per class.
- As already mentioned firing up an output routine without shipping out a page means that `TEX`'s top marks get wrong so it is impossible to rely on `TEX`'s approach directly. What we do instead is to keep track of the real marks (for the last page or more generally last region) in some global variables.
- These variables are updated in the output routine at defined places, i.e., when we do real output processing but not if we use special output routines to do internal housekeeping.
- The trick we use to get correctly updated variables is the following: the material that contains new marks (for example the page to be shipped out) is stored in a box. We then use `TEX` primitive box splitting functions by splitting off the largest amount possible (which should be the whole box if nothing goes really wrong). While that seems a rather pointless thing to do, it has one important side effect: `TEX` sets up first and bottom marks for each mark class from the material it has split off. This way we get the first and last marks (if there have been any) from the material in the box.
- The top marks are simply the last marks from the previous page or region. And if there hasn't been a first or bottom mark in the box then the new top mark also becomes new first and last mark for that class.
- That mark data is then stored in global token lists for use during the output routine and legacy commands such as `\leftmark` or new commands such as `\TopMark` simply access the data stored in these token lists.

That's about it in a nutshell. Of course, there are some details to be taken care of—those are discussed in the implementation sections.

6 Internal output routine functions

The functions in this section are tied to the output routine and used in the interface to L^AT_EX 2_E and perhaps at some later time within a new output routine for L^AT_EX. They are not meant for general use and are therefore made internal. Internal means that `@@` automatically gets replaced in the code (and in the documentation) so we have to give it a suitable value.

```
1  (@@=mark)
```

`__mark_update_singlecol_structures: __mark_update_singlecol_structures:`

L^AT_EX 2_E integration function in case we are doing single column layouts. It assumes that the page content is already stored in `\@outputbox` and processes the marks inside that box. It is called as part of `\@opcol`.

`__mark_update_dblcol_structures: __mark_update_singlecol_structures:`

L^AT_EX 2_E integration function mark used when we are doing double column documents. It assumes that the page content is already stored in `\@outputbox` and processes the marks inside that box. It then does different post-processing depending on the start of the switch `\if@firstcolumn`. If we are in the second column it also has to update page marks, otherwise it only updates column marks. It too is called as part of `\@opcol`.

`__mark_update_structure:nn __mark_update_structure:nn {<region>} {{material with marks}}`

Helper function that inspects the marks inside the second argument and assigns new mark values based on that to the `<region>` given in the first argument. For this it first copies the mark structure from `<region>` to `previous-<region>` and then takes all last mark values currently in the region and makes them the new top mark values. Finally it assigns new first and last values for all mark classes based on what was found in the second argument.

As a consequence, the allowed values for `<region>` are `page` and `column` because only they have `previous-...` counterparts.

Another important part to keep in mind is that marks are only recognized if they appear on top-level, e.g., if we want to process material stored in boxes we need to put it unboxed (using `\unvcopy` etc.) into the second argument.

`__mark_update_structure_alias:nn __mark_update_structure_alias:nn {<alias>} {{source}}`

Helper function that copies all mark values in the `<source>` region to `<alias>`, i.e., make the structures identical. Used to update the `previous-...` structures inside `__mark_update_structure:nn` and `first-column` and `last-column` structures inside `__mark_update_singlecol_structures:` or `__mark_update_dblcol_structures:`.

`__mark_update_structure_to_err:n __mark_update_structure_to_err:n {<region>}`

Helper function that sets all mark values in the `<region>` to an error message. This is currently used for `last-column` at times where using marks from it would be questionable/wrong, i.e., when we have just processed the first column in a two-column document.

7 The Implementation

```
2  {*2ekernel | latexrelease}
3  \ExplSyntaxOn
4  ⟨latexrelease⟩\NewModuleRelease{2022/06/01}{ltmarks}
5  ⟨latexrelease⟩          {Marks~handling}
```

7.1 Allocating new mark classes

`\g_mark_classes_seq` A list holding all the mark classes that have been declared.

```
6  \seq_new:N \g_mark_classes_seq
```

`\mark_new_class:n` A mark class is created by initializing a number of data structures. First, we get a register number to refer to the mark class. The new mark class is then added to the `\g_mark_classes_seq` sequence to be able to easily loop over all classes. Finally a number of top-level global token lists are declared that hold various versions of the mark for access.

```
7  \cs_new_protected:Npn \mark_new_class:n #1
8  {
9    \seq_if_in:NnTF \g_mark_classes_seq {#1}
10   {
11     \msg_error:nnn { mark } { class-already-defined }
12     {#1}
13   }
14   { \__mark_new_class:nn {#1} }
15 }
```

This is only available in the preamble.

```
16 \onlypreamble \mark_new_class:n
```

The internal command carries out the necessary allocations.

```
17 \cs_new_protected:Npn \__mark_new_class:nn #1
18 {
19 (*trace)
20   \__mark_debug:n { \iow_term:x { Marks:-new-mark:-#1-\msg_line_context: } }
21 (/trace)}
```

Use the $\text{\LaTeX}2\epsilon$ interface for now as the L3 programming layer doesn't have one for marks yet.

```
22 \exp_args:Nc \newmarks {c_mark_class_ #1 _mark}
```

Remember the new class in the sequence.

```
23 \seq_gput_right:Nn \g_mark_classes_seq {#1}
```

We need three token lists for each region, one for top, first, and last.

```
24 \tl_new:c { g_mark_page_top_ #1 _tl }
25 \tl_new:c { g_mark_page_first_ #1 _tl }
26 \tl_new:c { g_mark_page_last_ #1 _tl }
```

For the page region we also keep track of the previous-page.

```
27 \tl_new:c { g_mark_previous-page_top_ #1 _tl }
28 \tl_new:c { g_mark_previous-page_first_ #1 _tl }
29 \tl_new:c { g_mark_previous-page_last_ #1 _tl }
```

Same game for `column` and `previous-column`

```
30  \tl_new:c { g__mark_column_top_ #1 _tl }
31  \tl_new:c { g__mark_column_first_ #1 _tl }
32  \tl_new:c { g__mark_column_last_ #1 _tl }
33  \tl_new:c { g__mark_previous-column_top_ #1 _tl }
34  \tl_new:c { g__mark_previous-column_first_ #1 _tl }
35  \tl_new:c { g__mark_previous-column_last_ #1 _tl }
```

But for columns we also allocate token lists for the alias regions `first-column` and `last-column`.

```
36  \tl_new:c { g__mark_first-column_top_ #1 _tl }
37  \tl_new:c { g__mark_first-column_first_ #1 _tl }
38  \tl_new:c { g__mark_first-column_last_ #1 _tl }
39  \tl_new:c { g__mark_last-column_top_ #1 _tl }
40  \tl_new:c { g__mark_last-column_first_ #1 _tl }
41  \tl_new:c { g__mark_last-column_last_ #1 _tl }
42 }
```

(End of definition for `\mark_new_class:n` and `_mark_new_class:nn`. This function is documented on page 848.)

7.2 Updating mark structures

For some operations we need a temporary private box and two private global token lists.

```
43 \box_new:N \l_mark_box
44 \tl_new:N \g_mark_tmp_tl
45 \tl_new:N \g_mark_new_top_tl
```

(End of definition for `\l_mark_box`, `\g_mark_tmp_tl`, and `\g_mark_new_top_tl`.)

This function updates the mark structures. The first argument is the region to update and second argument receives the material that holds the marks. Out of this material we extract the first and last marks for all classes (if there are any) to do the assignments.

```
46 \cs_new_protected:Npn \_mark_update_structure:nn #1#2
47 {
```

First thing we do is copying the current structure to `previous-...`; this leaves the current structure untouched so we can update it class by class (which is necessary).

```
48     \_mark_update_structure_alias:nn { previous-#1 } {#1}
```

Getting the first and last marks out of the material in `#2` is done by putting the material in a box and then doing a split operation to the maximum size possible (which hopefully means all of the content).⁴¹ Because this is an action only for the sake of getting at the mark values we don't want any underfull box warnings so we turn those (locally) off.

```
49     \group_begin:
50         \dim_set_eq:NN \tex_splitmaxdepth:D \c_max_dim
51         \int_set_eq:NN \tex_vbadness:D \c_max_int
52         \dim_set_eq:NN \tex_vfuzz:D \c_max_dim
```

⁴¹We could verify this, maybe we should.

There is a further complication: if the region contains infinite shrinking glue then a `\vsplit` operation will balk with a low-level error. Now pages or columns, which are our main concern here, can't have such infinite shrinkage if they are cut straight from the galley, however the use of `\enlargethispage` actually does add some at the very bottom (and also wraps the whole page into a box by itself, so if we leave it this way then a) we get this error and b) we don't see any marks because they are hidden one level down).

Another possible issue are packages or user code that place stray `\vboxes` directly into the main galley (an example is `marginnote` that attaches its marginals in this way). If such boxes end up as the last item on the page we should not unpack them.

We therefore do an `\unskip` to get rid of that glue if present and also check if we have then a `\vbox` as the last item and if so unpack that too, but only under certain conditions, see below. All this is temporary, just for getting the marks out, so it doesn't affect the final page production.

In fact, we go one step further and set the box to a large negative height possible and afterwards take a look at the reported badness: if it is zero we know that there has still been infinite shrinkage in the box so that we can't do a `\vsplit`. If that is the case we generate an error message and bypass extracting the marks. We use only half of `\c_max_dim` because otherwise TeX will report an overfull vbox despite our setting of `\tex_vfuzz:D`. This test will not find existing infinite shrinkage in all cases, e.g., if there are several glues that cancel each other, but it is the best we can do.

```

53     \vbox_set_to_ht:Nnn \l_mark_box { -.5\c_max_dim }
54     {
55         #2
56         \tex_unskip:D
57         \box_set_to_last:N \l_mark_box

```

After having removed the last box from the current list (if there was one) we check if the list is now empty. If not, the the last box is definitely not the one from `\enlargethispage` and so we can and should leave it alone. Otherwise we check if this last box is a `\vbox`.

```

58     \int_compare:nNnT \tex_lastnodetype:D < 0
59     {
60         \box_if_vertical:NT \l_mark_box
61         {

```

If it is we do a further test and reset the `\l_mark_box` to check if it contains infinitely shrinkable glue.

```

62         \vbox_set_to_ht:Nnn \l_mark_box { -.5\c_max_dim }
63         {
64             \vbox_unpack:N \l_mark_box
65             \tex_kern:D \c_zero_dim % ensure that box
66                                         % is not empty
67         }

```

If not, then we unpack it, if yes we still ignore it for the process of mark extraction. We do not generate an error though, because in all likelihood this is an ordinary box like a marginal that does contain something like `\vss`.

```

68     \int_compare:nNnT \tex_badness:D > 0
69     {
70         \vbox_unpack:N \l_mark_box
71     }

```

If it wasn't a vbox, it was either an hbox or there was no box. Given that we are only interested in the marks we don't need put it back in that case. However, we have to

make sure that the outer box under construction is not totally empty (which it might have been from the start, or now), because TeX does not report a badness for empty boxes and that means our test would incorrectly conclude that we have infinite shrinking glue. A simple `\kern` is enough to avoid this (the same was already done above).

```

72          \tex_kern:D \c_zero_dim
73      }
74      \int_compare:nNnTF \tex_badness:D > 0

```

If the box had no infinite shrinkage (or rather if our test didn't show any) we vsplit it. Note that it doesn't matter that we set it to this strange size first. If there was infinite shrinkage after all, we end up with a low-level TeX error, but if there is, it is a coding error and needs correcting.

```

75      {
76          \vbox_set_split_to_ht:Nn \l_mark_box \l_mark_box \c_max_dim

```

After this action we can get first and last marks of the various classes through `\tex_splitfirstmarks:D` and `\tex_splitbotmarks:D`. So now we loop over all classes stored in `\g_mark_classes_seq`.

```

77      \seq_map_inline:Nn \g_mark_classes_seq
78      {

```

First action: get the last mark from the previous region, i.e., `previous-#1`. But because it is also still inside `#1`, at the moment we use that to construct the name because this is a tiny bit faster. Given that we need this value in various assignments we store it away which avoids unnecessary further csname generations.

```
    \tl_gset_eq:Nc \g_mark_new_top_tl { g_mark_#1_last_##1_tl }
```

This will first of all become the new top mark for the current class.

```
80      \tl_gset_eq:cN { g_mark_#1_top_##1_tl } \g_mark_new_top_tl
```

Next action is to get ourselves the new last mark from the material supplied.

```

81      \tl_gset:Nn \g_mark_tmp_tl
82          { \tex_splitbotmarks:D \use:c { c_mark_class_##1_mark } }

```

If this mark doesn't exist then obviously first mark does neither, so both become the last mark from the previous region. We have to be a little careful here: something like `\mark_insert:nn{foo}{}{}` adds an “empty” mark that should not be confused with no mark at all. But no mark in our material will result in `\g_mark_tmp_tl` being fully empty. This is why we have to make sure that “empty” from `\mark_insert:nn` only appears to be empty but fails the next test (see below how this is done).

```

83      \tl_if_empty:NTF \g_mark_tmp_tl
84          {
85              \tl_gset_eq:cN { g_mark_#1_last_ ##1_tl }
86                  \g_mark_new_top_tl
87              \tl_gset_eq:cN { g_mark_#1_first_##1_tl }
88                  \g_mark_new_top_tl
89          }

```

If it wasn't empty, i.e., if it had a real value then we use this value for our new last mark instead.

```

90      {
91          \tl_gset_eq:cN { g_mark_#1_last_##1_tl } \g_mark_tmp_tl

```

Because we had a last mark we also have a first mark (which might be the same, but might be not), so we pick that up and assign it to the appropriate token list. This explains why we first checked for the last mark because that makes the processing faster in case there is none.

```

92          \tl_gset:co { g__mark_##1_first_##1_tl }
93          {
94              \tex_splitfirstmarks:D
95                  \use:c { c__mark_class_##1_mark }
96          }
97      }
98  }
99 }
```

If the badness was zero (we actually tested for > 0 but it can't get negative) then we had infinite shrinkage, so we report that and set all marks to the value the last mark had before.

```

100  {
101      \msg_error:nnn { mark } { infinite-shrinkage } {#1}
102      \seq_map_inline:Nn \g__mark_classes_seq
103      {
104          \tl_gset_eq:cc { g__mark_##1_top_ ##1_tl }
105              { g__mark_##1_last_ ##1_tl }
106          \tl_gset_eq:cc { g__mark_##1_first_##1_tl }
107              { g__mark_##1_last_ ##1_tl }
108      }
109  }
```

Once all mark classes have been processed the data structures are updated and we can close the group which undoes our local changes and retains only the global ones.

```

110      \group_end:
111  }
```

(End of definition for __mark_update_structure:nn.)

__mark_update_structure_alias:nn

This function copies the structure for one region to another (name), e.g., from `page` to `previous-page` above, or later from `column` to `first-column`, etc.

```
112 \cs_new_protected:Npn \_\_mark_update_structure_alias:nn #1#2 {
```

This requires a simple loop through all mark classes copying the token list from one name to the next.

```

113  \seq_map_inline:Nn \g__mark_classes_seq
114  {
115      \tl_gset_eq:cc { g__mark_ #1_top_ ##1_tl }
116          { g__mark_ #2_top_ ##1_tl }
117      \tl_gset_eq:cc { g__mark_ #1_first_ ##1_tl }
118          { g__mark_ #2_first_ ##1_tl }
119      \tl_gset_eq:cc { g__mark_ #1_last_ ##1_tl }
120          { g__mark_ #2_last_ ##1_tl }
121  }
122 }
```

(End of definition for __mark_update_structure_alias:nn.)

```

\_\_mark_update_structure_to_err:n A slight variation is to install a fixed error message as the value.
\_\_mark_error:n
123 \cs_new_protected:Npn \_\_mark_update_structure_to_err:n #1 {
124   \seq_map_inline:Nn \g_\_\_mark_classes_seq
125   {
126     \tl_gset:cn { g_\_\_mark_ #1 _top_ } { \_\_mark_error:n {#1} }
127     \tl_gset:cn { g_\_\_mark_ #1 _first_ } { \_\_mark_error:n {#1} }
128     \tl_gset:cn { g_\_\_mark_ #1 _last_ } { \_\_mark_error:n {#1} }
129   }
130 }

```

Given that this is used in only one place, we could hardwire the argument which would be a bit more compact, but who knows, perhaps we end up with another reason to use this error command elsewhere, so for now we keep the argument.

```

131 \cs_new_protected:Npn \_\_mark_error:n #1 {
132   \msg_error:nnn { mark } { invalid-use } {#1}
133 }

```

(End of definition for `__mark_update_structure_to_err:n` and `__mark_error:n`.)

7.3 Placing and retrieving marks

`\mark_insert:nn` This function puts a mark for some $\langle class \rangle$ at the current point.

```

134 \cs_new_protected:Npn \mark_insert:nn #1#2
135 {
136   \seq_if_in:NnTF \g_\_\_mark_classes_seq {#1}
137   {

```

We need to pass the evaluated argument into the mark but protected commands should not expand including those protected using the `\protect` approach of L^AT_EX 2_E. We also disable `\label` and the like.⁴²

At this point the code eventually should get a public (and a kernel) hook instead of a set of hardwired settings.

```

138   \group_begin:

```

Within the group we alter some comments, e.g., `\label` or `\index`, to do the right at this point. This is done in the kernel hook `\@kernel@before@insertmark` which is followed by the public hook `insertmark` that can be used by packages to augment or alter that setup as necessary.

```

139           \@kernel@before@insertmark
140           \hook_use:n { insertmark }
141           \unrestored@protected@xdef \g_\_\_mark_tmp_tl {#2}
142   (*trace)
143     \_\_mark_debug:n{ \iow_term:x { Marks:~ set~#1~-<-
144       '\tl_to_str:V \g_\_\_mark_tmp_tl' ~ \msg_line_context: } }
145   (/trace)
146     \tex_marks:D \use:c { c_\_\_mark_class_ #1 _mark }
147   {

```

Here is the trick to avoid truly empty marks: if the result from the above processing is empty we add something which eventually becomes empty, but not immediately; otherwise we just put `\g___mark_tmp_tl` in.

```

148   \tl_if_empty:NTF \g_\_\_mark_tmp_tl

```

⁴²Straight copy from `latex.ltx` but is this even correct? At least a label in a running header makes little sense if it gets set several times! Maybe that needs looking at in the 2e kernel.

```

149          { \exp_not:n { \prg_do_nothing: } }
150          { \exp_not:o { \g__mark_tmp_tl } }
151      }
152  \group_end:

```

A mark introduces a possible break point and in certain situations that should not happen in vertical mode in L^AT_EX. This needs some cleanup . . .

```

153      \if@nobreak\ifvmode\nobreak\fi\fi
154  }

```

If the mark class was not known, raise an error.

```

155  {
156      \msg_error:nnx { mark } { unknown-class }
157      { \tl_to_str:n {#1} }
158  }
159 }

```

(End of definition for `\mark_insert:nn`. This function is documented on page 848.)

`\@kernel@before@insertmark` By default `\label`, `\index`, and `\glossary` do nothing when the mark is inserted.

```

insertmark
160 \cs_new:Npn \@kernel@before@insertmark {
161     \cs_set_eq:NN \label \scan_stop:
162     \cs_set_eq:NN \index \scan_stop:
163     \cs_set_eq:NN \glossary \scan_stop:
164 }

```

The public hook to augment the setup.

```

165 \hook_new:n {insertmark}

```

(End of definition for `\@kernel@before@insertmark` and `insertmark`.)

`\mark_use_top:nn`
`\mark_use_first:nn`
`\mark_use_last:nn`

To retrieve the first, last or top region mark, we grab the appropriate value stored in the corresponding token list variable and pass its contents back. These functions should be used only in output routines after `__mark_update_structure:nn` has acted, otherwise their value will be wrong.

If used with an unknown class or region they generate an error (fairly low-level because we are in an expandable context).

```

166 \cs_new:Npn \mark_use_first:nn #1#2 { \exp_not:v { \g__mark_#1_first_#2_tl } }
167 \cs_new:Npn \mark_use_last:nn #1#2 { \exp_not:v { \g__mark_#1_last_#2_tl } }
168 \cs_new:Npn \mark_use_top:nn #1#2 { \exp_not:v { \g__mark_#1_top_#2_tl } }

```

(End of definition for `\mark_use_top:nn`, `\mark_use_first:nn`, and `\mark_use_last:nn`. These functions are documented on page 849.)

7.4 Comparing mark values

`\mark_if_eq:nnnnTF` Test if in a given region (#1) for a given class (#2) the marks in position #3 and #4 (top, first, or last) are identical
`\mark_if_eq:nnnnnnTF`

```

169 \prg_new_conditional:Npnn \mark_if_eq:nnnn #1#2#3#4 { T , F , TF }
170 {
171     \tl_if_eq:ccTF { \g__mark_#1_#3_#2_tl }
172     { \g__mark_#1_#4_#2_tl }
173     \prg_return_true:
174     \prg_return_false:
175 }

```

The fully general test (with two triplets of the form $\langle region \rangle$, $\langle class \rangle$, and $\langle position \rangle$) is this:

```

176 \prg_new_conditional:Npnn \mark_if_eq:nnnnnn #1#2#3#4#5#6 { T , F , TF }
177 {
178   \tl_if_eq:ccTF { g__mark_ #1 _#3_ #2 _tl }
179   { g__mark_ #4 _#6_ #5 _tl }
180   \prg_return_true:
181   \prg_return_false:
182 }
```

(End of definition for `\mark_if_eq:nnnnTF` and `\mark_if_eq:nnnnnnTF`. These functions are documented on page 850.)

7.5 Messages

Mark errors are LaTeX kernel errors:

```

183 \prop_gput:Nnn \g_msg_module_type_prop { mark } { LaTeX }
184 \msg_new:nnn { mark } { class-already-defined }
185 { Mark~class~'#1'~already~defined }
186 {
187   \c__msg_coding_error_text_tl
188   LaTeX~was~asked~to~define~a~new~mark~class~called~'#1';~
189   this~mark~class~already~exists.
190   \c__msg_return_text_tl
191 }

192 \msg_new:nnn { mark } { unknown-class }
193 { Unknown~mark~class~'#1'. }
194 {
195   \c__msg_coding_error_text_tl
196   LaTeX~was~asked~to~manipulate~a~mark~of~class~'#1',~
197   but~this~class~of~marks~does~not~exist.
198 }

199
200 \msg_new:nnn { mark } { invalid-use }
201 { Mark~region~'#1'~not ~usable }
202 {
203   \c__msg_coding_error_text_tl
204   The~region~'#1'~can~only~be~used~after~
205   all~columns~have~been~assembled.
206   \c__msg_return_text_tl
207 }

208 \msg_new:nnn { mark } { infinite-shrinkage }
209 { Infinite~shrinkage~found~in~'#1'. }
210 {
211   \c__msg_coding_error_text_tl
212   The~mark~region~'#1'~contains~some~infinite~negative~glue~
213   allowing~it~to~shrink~to~an~arbitrary~size.~
214   This~makes~it~impossible~to~split~the~region~apart~to~
215   get~at~its~marks.~They~are~lost.
216 }
```

7.6 Debugging the mark structures

Code and commands in this section are not final, it needs more experimentation to see what kind of tracing information is going to be useful in practice. For now the tracing is mainly meant to be used for code testing and not so much for application testing.

It is quite likely that the commands and the behavior of the tracing might change in the future once we gained some experience with it.

\g_mark_debug_bool Holds the current debugging state.

217 \bool_new:N \g_mark_debug_bool

(End of definition for \g_mark_debug_bool.)

\mark_debug_on: Turns debugging on and off by redefining __mark_debug:n.

\mark_debug_off:

__mark_debug:n \cs_new_eq:NN __mark_debug:n \use_none:n

219 \cs_new_protected:Npn \mark_debug_on:

220 {

221 \bool_gset_true:N \g_mark_debug_bool

222 __mark_debug_gset:

223 }

224 \cs_new_protected:Npn \mark_debug_off:

225 {

226 \bool_gset_false:N \g_mark_debug_bool

227 __mark_debug_gset:

228 }

229 \cs_new_protected:Npn __mark_debug_gset:

230 {

231 \cs_gset_protected:Npx __mark_debug:n ##1

232 { \bool_if:NT \g_mark_debug_bool {##1} }

233 }

(End of definition for \mark_debug_on: and others. These functions are documented on page 851.)

\DebugMarksOn CamelCase commands for debugging.

\DebugMarksOff

234 \cs_new_eq:NN \DebugMarksOn \mark_debug_on:

235 \cs_new_eq:NN \DebugMarksOff \mark_debug_off:

(End of definition for \DebugMarksOn and \DebugMarksOff. These functions are documented on page 851.)

__mark_class_status:nn Shows the mark values across all regions for one mark class (#2). The first argument gives some <info> to help identifying where the command was called.

236 {*trace}

237 \cs_new_protected:Npn __mark_class_status:nn #1#2

238 {

239 \typeout{ Marks:#2~ #1:}

240 \typeout{@spaces page~ (current):

241 | \exp_not:v { g_mark_page_top_ #2 _tl }

242 | \exp_not:v { g_mark_page_first_ #2 _tl }

243 | \exp_not:v { g_mark_page_last_ #2 _tl } |}

244 \typeout{@spaces page~ (previous):

245 | \exp_not:v { g_mark_previous-page_top_ #2 _tl }

246 | \exp_not:v { g_mark_previous-page_first_ #2 _tl }

247 | \exp_not:v { g_mark_previous-page_last_ #2 _tl } |}

248 \typeout{@spaces column~ (previous):

```

249 | \exp_not:v { g__mark_previous-column_top_ #2 _tl }
250 | \exp_not:v { g__mark_previous-column_first_ #2 _tl }
251 | \exp_not:v { g__mark_previous-column_last_ #2 _tl }     |}
252 \typeout{\@spaces column~ (current):
253 | \exp_not:v { g__mark_column_top_ #2 _tl }
254 | \exp_not:v { g__mark_column_first_ #2 _tl }
255 | \exp_not:v { g__mark_column_last_ #2 _tl }     |}
256 \typeout{\@spaces column~ (first):
257 | \exp_not:v { g__mark_first-column_top_ #2 _tl }
258 | \exp_not:v { g__mark_first-column_first_ #2 _tl }
259 | \exp_not:v { g__mark_first-column_last_ #2 _tl }     |}
260 \typeout{\@spaces column~ (second):
261 | \exp_not:v { g__mark_last-column_top_ #2 _tl }
262 | \exp_not:v { g__mark_last-column_first_ #2 _tl }
263 | \exp_not:v { g__mark_last-column_last_ #2 _tl }     |}
264 }

```

(End of definition for `_mark_class_status:nn`.)

`_mark_status:n` Show all mark class values across all regions.

```

265 \cs_new_protected:Npn \_mark_status:n #1
266 {
267   \seq_map_inline:Nn \g__mark_classes_seq
268   { \_mark_class_status:nn {#1} {##1} }
269 }
270 
```

(End of definition for `_mark_status:n`.)

7.7 Designer-level interfaces

`\NewMarkClass` These two are identical to the L3 programming layer commands.

```

271 \cs_new_eq:NN \NewMarkClass \mark_new_class:n
272 \onlypreamble \NewMarkClass
273 \cs_new_eq:NN \InsertMark \mark_insert:nn

```

(End of definition for `\NewMarkClass` and `\InsertMark`. These functions are documented on page 848.)

`\TopMark` The following commands take an optional argument that defaults to page. There is no checking that the region is actually valid. If not there is simply an empty return.

```

274 \NewExpandableDocumentCommand \FirstMark { O{page} m }
275           { \mark_use_first:nn {#1}{#2} }
276 \NewExpandableDocumentCommand \LastMark { O{page} m }
277           { \mark_use_last:nn {#1}{#2} }
278 \NewExpandableDocumentCommand \TopMark { O{page} m }
279           { \mark_use_top:nn {#1}{#2} }

```

(End of definition for `\TopMark`, `\FirstMark`, and `\LastMark`. These functions are documented on page 849.)

\IfMarksEqualTF We only provide a CamelCase command for the case with one region (optional) and one class. One could think of also providing a version for the general case with several optional arguments, but use cases for this are most likely rare, so not done yet.

```
280 \NewExpandableDocumentCommand \IfMarksEqualTF {0{page}mmm} {
281   \mark_if_eq:nnnnTF {#1}{#2}{#3}{#4}
282 }
```

(End of definition for \IfMarksEqualTF. This function is documented on page 850.)

8 L^AT_EX 2 _{ϵ} integration

8.1 Core L^AT_EX 2 _{ϵ} integration

_mark_update_singlecol_structures: This command updates the mark structures if we are producing a single column document.

```
283 \cs_new_protected:Npn \_mark_update_singlecol_structures: {
```

First we update the page region (which also updates the previous-page).

The \outputbox is normally in \vbox in L^AT_EX but we can't take that for granted (an amsmath test document changed it to an \hbox just to trip me up) so we are a little careful with unpack now.

```
284 \box_if_vertical:NTF \outputbox
285   {
286     \_mark_update_structure:nn {page}
287     { \vbox_unpack:N \outputbox }
288   }
289   {
290     \_mark_update_structure:nn {page}
291     { \hbox_unpack:N \outputbox }
292   }
```

The we provide the necessary updates for the aliases.

```
293 \_mark_update_structure_alias:nn {previous-column}{previous-page}
294 \_mark_update_structure_alias:nn {column}{page}
295 \_mark_update_structure_alias:nn {first-column}{page}
296 \_mark_update_structure_alias:nn {last-column}{page}
297 (*trace)
298 % move this into status itself?
299   \_mark_debug:n
300   {
301     \_mark_status:n
302     { in~ OR~ (
303       \legacy_if:nTF {@twoside}
304       { twoside-
305         \int_if_odd:nTF \c@page
306         { odd }{ even }
307       }
308       { oneside }
309     )
310   }
311 }
312 (*/trace)
313 }
```

(End of definition for `__mark_update_singlecol_structures`.)

`__mark_update_dblcol_structures`: This command handles the updates if we are doing two-column pages.

314 `\cs_new_protected:Npn __mark_update_dblcol_structures: {`

First we update the `column` and `previous-column` regions using the material assembled in `\@outputbox`.

```
315   \box_if_vertical:NTF \@outputbox
316    {
317      \__mark_update_structure:nn {column}
318      { \vbox_unpack:N \@outputbox }
319    }
320    {
321      \__mark_update_structure:nn {column}
322      { \hbox_unpack:N \@outputbox }
323  }
```

How we have to update the alias regions depends on whether or not `\copcol` was called to process the first column or to produce the completed page

```
324   \legacy_if:nTF {@firstcolumn}
325    {
```

If we are processing the first column then `column` is our `first-column` and there is no `last-column` yet, so we make those an error.

```
326      \__mark_update_structure_alias:nn {first-column}{column}
327      \__mark_update_structure_to_err:n {last-column}
328  }
329  {
```

If we produce the completed page then the `first-column` is the same as the new `previous-column`. However, the structure should already be correct if you think about it (because it was set to `column` last time which is now the `previous-column`), thus there is no need to make an update.

```
330 %   \__mark_update_structure_alias:nn {first-column}{previous-column}
```

However, we now have a proper `last-column` so we assign that.

```
331   \__mark_update_structure_alias:nn {last-column}{column}
```

What now remains doing is to update the `page` and `previous-page` regions. For this we have to copy the settings in `page` into `previous-page` and then update `page` such that the top and first marks are taken from the `first-column` region and the last marks are taken from the `last-column` region. All this has to be done for all mark classes so we loop over our sequence.

Note that one loop is needed if we arrange the copy statements in a suitable way.

```
332   \seq_map_inline:Nn \g__mark_classes_seq
333    {
334      \tl_gset_eq:cc { g__mark_previous-page_top_ ##1 _tl }
335      { g__mark_page_top_ ##1 _tl }
336      \tl_gset_eq:cc { g__mark_previous-page_first_ ##1 _tl }
337      { g__mark_page_first_ ##1 _tl }
338      \tl_gset_eq:cc { g__mark_previous-page_last_ ##1 _tl }
339      { g__mark_page_last_ ##1 _tl }
```

The `page` updates need to come after the corresponding updates for `previous-page` otherwise we loose the necessary value.

```

340          \tl_gset_eq:cc { g__mark_page_top_      ##1 _tl }
341          { g__mark_first-column_top_  ##1 _tl }
342          \tl_gset_eq:cc { g__mark_page_first_   ##1 _tl }
343          { g__mark_first-column_first_ ##1 _tl }
344          \tl_gset_eq:cc { g__mark_page_last_    ##1 _tl }
345          { g__mark_last-column_last_  ##1 _tl }
346      }
347  }
348 (*trace)
349     \__mark_debug:n
350     {
351         \__mark_status:n
352         { in~ OR~ (
353             \legacy_if:nTF {@twoside}
354             { twoside-
355                 \int_if_odd:nTF \c@page
356                 { odd }{ even }
357             }
358             { oneside }
359             \space
360             \legacy_if:nTF {@firstcolumn}
361             { first~ }{ second~ }
362             column )
363         }
364     }
365 (/trace)
366 }
```

(End of definition for `__mark_update_dblcol_structures:..`)

367 `\@=`

```
@expl0@@mark@update@singlecol@structures@@
@expl0@@mark@update@dblcol@structures@@
368 \cs_new_eq:NN  \@expl0@@mark@update@singlecol@structures@@
369           \__mark_update_singlecol_structures:
370 \cs_new_eq:NN  \@expl0@@mark@update@dblcol@structures@@
371           \__mark_update_dblcol_structures:
```

(End of definition for `\@expl0@@mark@update@singlecol@structures@@` and
`\@expl0@@mark@update@dblcol@structures@@`.)

8.2 Other L^AT_EX 2 _{ϵ} output routines

This section will cover `multicol` and other packages altering or providing their own output routine. Not done yet.

```

372 \if@latexrelease\IncludeInRelease{0000/00/00}{ltmarks}%
373 \else\fi
374 
```

We keep the interface commands around even if we roll back in case they are used in packages that don't roll back. Not likely to do a lot of good, but then there is not much we can do, but this at least then doesn't give errors.

```
375 〈\latexrelease〉\DeclareRobustCommand \NewMarkClass[1]{}
376 〈\latexrelease〉\DeclareRobustCommand \InsertMark[2]{}
377 〈\latexrelease〉\RenewExpandableDocumentCommand \FirstMark { O{} m } { }
378 〈\latexrelease〉\RenewExpandableDocumentCommand \LastMark { O{} m } { }
379 〈\latexrelease〉\RenewExpandableDocumentCommand \TopMark { O{} m } { }
380 〈\latexrelease〉\RenewExpandableDocumentCommand \IfMarksEqualTF { O{} mmm }{ }
381 〈\latexrelease〉
```

Same here, this avoided extra roll back code in the OR.

```
382 〈\latexrelease〉\let \Expl@@mark@update@singlecol@structures@@ \relax
383 〈\latexrelease〉\let \Expl@@mark@update@dblcol@structures@@ \relax
384 〈\latexrelease〉
385 〈\latexrelease〉
386 〈\latexrelease〉\EndModuleRelease
387 \ExplSyntaxOff
388 〈/2ekernel | \latexrelease〉
      Reset module prefix:
389 〈@@=〉
```

File T

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 of 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 of 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 of 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 of 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 of definition for \@leftmark and \@rightmark.)

```
18 {/2ekernel}
19 {*2ekernel | latexrelease}
20 {latexrelease}\IncludeInRelease{2022/06/01}%
21 {latexrelease} {\markboth}{New mark support}%
```

\markboth User commands for setting L^AT_EX 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 \ExplSyntaxOn
23 \DeclareRobustCommand*\markboth[2]{%
24   \begingroup
25   \let\label\relax \let\index\relax \let\glossary\relax
26   \unrestored@protected\xdef\@themark {{\#1}{\#2}}%
27   \temptokena\expandafter{\@themark}}
```

In addition to generating the legacy mark we output the individual ones as well at the very same point. The legacy mark is kept unchanged in order to work with packages that expect that mark in exactly the way it is right now.

We might want to think about how to improve this in one-side documents, see comments below.

We have not changed all of the code to L3 prog layer convention, in case packages attempt to do some patching and expect the 2e names being around. Eventually this should and will change.

```
28 \mark_insert:nn{2e-left}{\#1}
29 \mark_insert:nn{2e-right}{\#2}
```

```

30      \tl_if_empty:nF{#2}{ \mark_insert:nn{2e-right-nonempty}{#2} }
31      \mark{\the\@temptokena}%
32  \endgroup
33  \if@nobreak\ifvmode\nobreak\fi\fi}

34 \DeclareRobustCommand*\markright[1]{%
35   \begingroup
36     \let\label\relax \let\index\relax \let\glossary\relax

```

Protection is handled inside \markright.

```

37   \expandafter\markright\@themark {#1}%
38   \@temptokena \expandafter{\@themark}%

```

Same game with \markright more or less ...

```

39   \mark_insert:nn{2e-right}{#1}
40   \tl_if_empty:nF{#1}{ \mark_insert:nn{2e-right-nonempty}{#1} }

```

The legacy L^AT_EX mechanism always sets left and right mark, i.e., if a sub-mark (i.e., right mark) is set the corresponding main mark also is getting a mark with the same value it had previously. However, for the individual mark classes this means we are losing information so for them that is not done.

```

41 %   \mark_insert:nn{2e-left}{\exp_after:wN \use_i:nn \@themark }
42   \mark{\the\@temptokena}%
43  \endgroup
44  \if@nobreak\ifvmode\nobreak\fi\fi}
45 \ExplSyntaxOff

```

(End of definition for \markboth and \markright. These functions are documented on page 851.)

```

46 </2ekernel | latexrelease>
47 <latexrelease>\EndIncludeInRelease
48 <latexrelease>\IncludeInRelease{2019/10/01}%
49 <latexrelease>           {\markboth}{Make commands robust}%
50 <latexrelease>
51 <latexrelease>\DeclareRobustCommand*\markboth[2]{%
52 <latexrelease> \begingroup
53 <latexrelease>   \let\label\relax \let\index\relax \let\glossary\relax
54 <latexrelease>   \unrestored@protected@xdef@\themark {{#1}{#2}}%
55 <latexrelease>   \@temptokena \expandafter{\@themark}%
56 <latexrelease>   \mark{\the\@temptokena}%
57 <latexrelease> \endgroup
58 <latexrelease> \if@nobreak\ifvmode\nobreak\fi\fi}
59 <latexrelease>\DeclareRobustCommand*\markright[1]{%
60 <latexrelease> \begingroup
61 <latexrelease>   \let\label\relax \let\index\relax \let\glossary\relax
62 <latexrelease>   \expandafter\markright\@themark {#1}%
63 <latexrelease>   \@temptokena \expandafter{\@themark}%
64 <latexrelease>   \mark{\the\@temptokena}%
65 <latexrelease> \endgroup
66 <latexrelease> \if@nobreak\ifvmode\nobreak\fi\fi}
67 <latexrelease>
68 <latexrelease>\EndIncludeInRelease
69 <latexrelease>\IncludeInRelease{0000/00/00}%
70 <latexrelease>           {\markboth}{Make commands robust}%
71 <latexrelease>
72 <latexrelease>\kernel@make@fragile\markboth

```

```

73  ⟨latexrelease⟩\kernel@make@fragile\markright
74  ⟨latexrelease⟩
75  ⟨latexrelease⟩\EndIncludeInRelease
76  ⟨*2ekernel⟩

\@markright
\leftmark
\rightmark
77 \def\@markright#1#2#3{\@temptokena {#1}%
78   \unrestored@protected@xdef\@themark{{\the\@temptokena}{#3}}}
79 \def\leftmark{\expandafter\@leftmark\botmark\@empty\@empty}
80 \def\rightmark{\expandafter\@rightmark\firstmark\@empty\@empty}

(End of definition for \@markright, \leftmark, and \rightmark.)

\@themark Initialise LATEX's marks without setting a TEX mark ⟨whatsit⟩.
81 \def\@themark{}{ }

(End of definition for \@themark.)

\mark Test versions of LATEX 2E initialised TEX's \mark system at this point, but this was
removed before the first release.

AtBeginDocument{\mark{}{}}

(End of 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.)
82 \DeclareRobustCommand\raggedbottom{%
83   \def\@textbottom{\vskip \z@ \oplus.0001fil}\let\@texttop\relax}

(End of definition for \raggedbottom.)

\flushbottom \flushbottom: Inverse of \raggedbottom — makes all pages the same height.
84 \DeclareRobustCommand\flushbottom{%
85   \let\@textbottom\relax \let\@texttop\relax}

(End of definition for \flushbottom.)

\sloppy \sloppy will never (well, hardly ever) produce overfull boxes, but may produce underfull
ones. (14 June 85)
86 \DeclareRobustCommand\sloppy{%
87   \tolerance 9999%
88   \emergencystretch 3em%
89   \hfuzz .5\p@
90   \vfuzz\hfuzz}

(End of definition for \sloppy.)

\slloppypar (env.) A slloppypar environment is equivalent to {\par \sloppy ... \par}.
91 \def\slloppypar{\par\sloppy}
92 \def\endslloppypar{\par}

```

\fussy Resets T_EX's parameters to their normal finicky values.

```
93 \DeclareRobustCommand\fussy{%
94   \emergencystretch\z@
95   \tolerance 200%
96   \hfuzz .1\p@
97   \vfuzz\hfuzz}
```

(End of definition for **\fussy**.)

\overfullrule L^AT_EX default is no overfull box rule. Changed by document class option.

```
98 \overfullrule 0pt
```

(End of definition for **\overfullrule**.)

```
99 </2ekernel>
```

File U

ltclass.dtx

1 Introduction

This file implements the following declarations, which replace `\documentstyle` in L^AT_EX 2_E 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^AT_EX 2_E and L^AT_EX 2.09 is that L^AT_EX 2_E 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^AT_EX 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^AT_EX 2.09 compatibility mode. As far as most packages are concerned, this only affects the warnings and errors L^AT_EX generates. This flag does affect the definition of font commands, and `\sloppy`.

`\usepackage[⟨package-option-list⟩]{⟨package-list⟩}[⟨version⟩]`

There can be any number of these declarations. All packages in *⟨package-list⟩* are called with the same options.

Each *⟨package⟩* file defines new elements (or modifies those defined in the *⟨class⟩*), and thus extends the range of documents which can be processed. The *⟨package-option-list⟩* is a list of options which can modify the formatting of elements defined in the *⟨package⟩* file. The *⟨version⟩* is a version number, beginning with a date in the format YYYY/MM/DD. If an older version of the package is found, a warning is issued.

Each package is loaded only once. If the same package is requested more than once, nothing happens, unless the package has been requested with options that were not given the first time it was loaded, in which case an error is produced.

As well as processing the options given in the *⟨package-option-list⟩*, each package processes the *⟨main-option-list⟩*. This means that options that affect all of the packages can be given globally, rather than repeated for every package.

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

`filecontents` (*env.*)

The environment `filecontents` is intended for passing the contents of packages, options, or other files along with a document in a single file. It has one argument, which is the name of the file to create. If that file already exists (maybe only in the current directory if the OS supports a notion of a ‘current directory’ or ‘default directory’) then nothing happens (except for an information message) and the body of the environment is bypassed. Otherwise, the body of the environment is written verbatim to the file name given as the first argument, together with some comments about how it was produced.

The environment can also be called with an optional argument which is used to alter some of its behavior: option `force` or `overwrite` will allow for overwriting existing files, option `nosearch` will only check the current directory when looking if the file exists. This can be useful if you want to generate a local (modified) copy of some file that is already in the search tree of TeX. Finally, you can use `noheader` to prevent it from writing the standard blurb at the top of the file (this is actually the same as using the star form of the environment).

The environment is now allowed anywhere in the document, but to ensure that all packages or options necessary are available when the document is run, it is normally best to place it at the top of your file (before `\documentclass`). A possible use case for using it inside the document body is if you want to reuse some text several times in the document you could then write it and later use `\input` to retrieve it where needed.

The begin and end tags should each be on a line by itself.

2.1 Option processing

When the options are processed, they are divided into two types: *local* and *global*:

- For a class, the options in the `\documentclass` command are local.
- For a package, the options in the `\usepackage` command are local, and the options in the `\documentclass` command are global.

The options for `\documentclass` and `\usepackage` are processed in the following way:

1. The local and global options that have been declared (using `\DeclareOption` as described below) are processed first.

In the case of `\ProcessOptions`, they are processed in the order that they were declared in the class or package.

In the case of `\ProcessOptions*`, they are processed in the order that they appear in the option-lists. First the global options, and then the local ones.

2. Any remaining local options are dealt with using the default option (declared using the `\DeclareOption*` declaration described below). For document classes, this usually does nothing, but records the option on a list of unused options. For packages, this usually produces an error.

Finally, when `\begin{document}` is reached, if there are any global options which have not been used by either the class or any package, the system will produce a warning.

3 Class and Package interface

3.1 Class name and version

`\ProvidesClass` A class can identify itself with the `\ProvidesClass{name}[version]` command. The *version* should begin with a date in the format YYYY/MM/DD.

3.2 Package name and version

`\ProvidesPackage` A package can identify itself with the `\ProvidesPackage{name}[version]` command. The *version* should begin with a date in the format YYYY/MM/DD.

3.3 Requiring other packages

`\RequirePackage` Packages or classes can load other packages using `\RequirePackage[options]{name}[version].`

If the package has already been loaded, then nothing happens unless the requested options are not a subset of the options with which it was loaded, in which case an error is called.

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

`\PassOptionsToPackage` Packages can pass options to other packages using:

`\PassOptionsToPackage{options}{package}`.

`\PassOptionsToClass` This adds the *options* to the options list of any future `\RequirePackage` or `\usepackage` command. For example:

```
\PassOptionsToPackage{foo,bar}{fred}
```

is the same as:

```
\RequirePackage[foo,bar,baz]{fred}
```

`\LoadClassWithOptions` `\LoadClassWithOptions{name}[version]:`

This is similar to `\LoadClass`, but it always calls class *name* with exactly the same option list that is being used by the current class, rather than an option explicitly supplied or passed on by `\PassOptionsToClass`. `\RequirePackageWithOptions` is the analogous command for packages.

This is mainly intended to allow one class to simply build on another, for example:

```
\LoadClassWithOptions{article}
```

This should be contrasted with the slightly different construction

```
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions
\LoadClass{article}
```

As used here, the effects are more or less the same, but the version using `\LoadClassWithOptions` is slightly quicker (and less to type). If, however, the class declares options of its own then the two constructions are different; compare, for example:

```
\DeclareOption{landscape}{...}
\ProcessOptions
\LoadClassWithOptions{article}
```

with:

```
\DeclareOption{landscape}{...}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions
\LoadClass{article}
```

In the first case, the `article` class will be called with option `landscape` precisely when the current class is called with this option; but in the second example it will not as in that case `article` is only passed options by the default option handler, which is not used for `landscape` as that option is explicitly declared.

`\IfPackageLoadedTF` To find out if a package has already been loaded, use
`\IfClassLoadedTF` `\IfPackageLoadedTF{(package)}{<true>}{<false>}`
`\@ifpackageloaded`
`\@ifclassloaded`
or the old name `\@ifpackageloaded`.
`\IfPackageAtLeastTF` To find out if a package has already been loaded with a version equal to or more recent than `<date>`, use
`\IfClassAtLeastTF` `\IfPackageAtLeastTF{(package)}{<date>}{{<true>}{<false>}}`
`\@ifpackagelater`
`\@ifclasslater` or the old name `\@ifpackagelater`.
`\IfFormatAtLeastTF` To test the format date use

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

`\IfPackageLoadedWithOptionsTF` To find out if a package has already been loaded with at least the options `<options>`,
`\IfClassLoadedWithOptionsTF` use
`\@ifpackagewith` `\IfPackageLoadedWithOptionsTF{(package)}{<options>}{{<true>}{<false>}}`
`\@ifclasswith`
or the old name `\@ifpackagewith`.
There exists one package that can't be tested with the above commands: the `fontenc` package pretends that it was never loaded to allow for repeated reloading with different options (see `ltoutenc.dtx` for details).

3.4 Declaring new options

Options for classes and packages are built using the same macros.

`\DeclareOption` To define a builtin option, use `\DeclareOption{<name>}{{<code>}}`.
`\DeclareOption*` To define the default action to perform for local options which have not been declared, use `\DeclareOption*{{<code>}}`.
Note: there should be no use of
`\RequirePackage`, `\DeclareOption`, `\DeclareOption*` or `\ProcessOptions` inside `\DeclareOption` or `\DeclareOption*`.
Possible uses for `\DeclareOption*` include:
`\DeclareOption*{}`
Do nothing. Silently accept unknown options. (This suppresses the usual warnings.)
`\DeclareOption*{\@unkownoptionerror}`
Complain about unknown local options. (The initial setting for package files.)
`\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{<pkg-name>}}`
Handle the current option by passing it on to the package `<pkg-name>`, which will presumably be loaded via `\RequirePackage` later in the file. This is useful for building

‘extension’ packages, that perhaps handle a couple of new options, but then pass everything else on to an existing package.

```
\DeclareOption*{\InputIfFileExists{xx-\CurrentOption.yyy}%
{}%
{\OptionNotUsed}}
```

Handle the option `foo` by loading the file `xx-foo.yyy` if it exists, otherwise do nothing, but declare that the option was not used. Actually the `\OptionNotUsed` declaration is only needed if this is being used in class files, but does no harm in package files.

3.5 Safe Input Macros

<code>\InputIfFileExists</code>	<code>\InputIfFileExists{\langle file \rangle}{\langle then \rangle}{\langle else \rangle}</code>
	Inputs <code>\langle file \rangle</code> if it exists. Immediately before the input, <code>\langle then \rangle</code> is executed. Otherwise <code>\langle else \rangle</code> is executed.
<code>\IfExists</code>	As above, but does not input the file.
	One thing you might like to put in the <code>\langle else \rangle</code> clause is
<code>\@missingfileerror</code>	This starts an interactive request for a filename, supplying default extensions. Just hitting return causes the whole input to be skipped and entering <code>x</code> quits the current run,
<code>\input</code>	This has been redefined from the L ^A T _E 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 of 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 of 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 of definition for \@declaredoptions.)

\@classoptionslist List of options of the main class.

```
 9  \let\@classoptionslist\relax  
10  \%@\onlypreamble\@classoptionslist
```

(End of definition for \@classoptionslist.)

\@raw@classoptionslist List of options of the main class (unprocessed).

```
11  \let\@raw@classoptionslist\relax
```

(End of 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 of definition for \@unusedoptionlist.)

\CurrentOption Name of current package or option.

```
14  \let\CurrentOption\@empty
```

(End of 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 of definition for \@currpath.)

\@currname Name of current package or option.

```
29  \let\@currname\@empty
```

(End of definition for \@currname.)

\@currext The current file extension.

```
30  \global\let\@currext=\@empty
```

(End of definition for \@currext.)

\@clsextension The two possible values of \@currext.

\@pkgextension
31 \def\@clsextension{cls}
32 \def\@pkgextension{sty}

(End of definition for \cclsextension and \pkgextension.)

```
\@pushfilename Commands to push and pop the file name and extension.  
\@popfilename #1 current name.  
\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}{\@pushfilename}%  
37 (latexrelease) {Add \@expl@push@filename@@ and \@expl@push@filename@aux@@}%  
38 \def\@pushfilename{%
```

The push and pop macros are injected in \@pushfilename and \@popfilename so that they correctly keep track of the hook labels.

This needs cleanup with the expl3 interfaces also playing here, e.g., \@expl@push@filename@@ needs cleanup and (and should probably not have this name either).

```
39  \@expl@push@filename@@  
40  \xdef\currnamestack{  
41   {\@currname}%  
42   {\@currext}%  
43   {\the\catcode`@}%  
44   \currnamestack}%
```

Temporarily add a stack for \currpath here. This should be integrated in the main file stack eventually, but other packages rely on \currnamestack having three elements per file, so that isn't a trivial change. The prefix \@kernel@... hopefully discourages people from using it.

```
45  \xdef\kernel@currpathstack{  
46  {\currpath}%  
47  \kernel@currpathstack}%  
48  \@expl@push@filename@aux@@}  
49 (latexrelease)\EndIncludeInRelease
```

The following version of \@pushfilename didn't formally exist in this file, but in the 2020/02/02 release, expl3 was preloaded and it patched \@pushfilename (and \@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 \@popfilename.

```
50 (latexrelease)  
51 (latexrelease)\IncludeInRelease{2020/02/02}{\@pushfilename}%  
52 (latexrelease) {Add \@expl@push@filename@@}%  
53 (latexrelease)\def\@pushfilename{  
54 (latexrelease) \@expl@push@filename@@  
55 (latexrelease) \xdef\currnamestack{  
56 (latexrelease) {\@currname}%  
57 (latexrelease) {\@currext}%  
58 (latexrelease) {\the\catcode`@}%  
59 (latexrelease) \currnamestack}%  
60 (latexrelease) \@expl@push@filename@aux@@}  
61 (latexrelease)\EndIncludeInRelease  
62 (latexrelease)
```

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^AT_EX > 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}
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 of 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  <latexrelease> {Add \@kernel@currpathstack}%
146  <latexrelease>\let\@kernel@currpathstack\@undefined
147  <latexrelease>\let\@p@filepath\@undefined
148  <latexrelease>\let\@p@filepath@aux\@undefined
149  <latexrelease>\EndIncludeInRelease
150  </2ekernel | latexrelease>
151  {*2ekernel}

```

(End of 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 of definition for \@optionlist.)

\@ifpackageloaded \@ifpackageloaded{<name>} Checks to see whether a file has been loaded.

```

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 of 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 of 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.

```

165  </2ekernel>
166  {*2ekernel | latexrelease>
167  <latexrelease>\IncludeInRelease{2020/10/01}%
168  <latexrelease>          {\IfFormatAtLeastTF}{Test format date}%
169  \def\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
170  \let\IfPackageAtLeastTF\@ifpackagelater
171  \let\IfClassAtLeastTF\@ifclasslater
172  \def\IfFileAtLeastTF#1{\expandafter\@ifl@t@r\csname ver@#1\endcsname}

```

For rollback pretend it was available since the beginning of dawn.

```

173  </2ekernel | latexrelease>
174  <latexrelease>\EndIncludeInRelease
175  <latexrelease>\IncludeInRelease{0000/00/00}%
176  <latexrelease>          {\IfFormatAtLeastTF}{Test format date}%
177  <latexrelease>\def\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
178  <latexrelease>\let\IfPackageAtLeastTF\@ifpackagelater

```

```

179  \let\IfClassAtLeastTF\@ifclasslater
180  \def\IfFileAtLeastTF#1{\expandafter\@ifl@t@r\csname ver@#1\endcsname}
181  \EndIncludeInRelease
182  \end{ekernel}

```

(End of definition for `\IfPackageAtLeastTF` and others.)

`\@ifl@ter`

```

183  \def\@ifl@ter#1#2{%
184    \expandafter\@ifl@t@r
185      \csname ver@#2.#1\endcsname}
186  \end{ekernel}

This internal macro is also used in \NeedsTeXFormat.
187  \IncludeInRelease[2018/04/01]%
188  {\@ifl@t@r}{Guard against bad input}%
189  \end{ekernel | latexrelease}
190  \def\@ifl@t@r#1#2{%
191    \ifnum\expandafter\@parse@version@#1//00\@nil<%
192      \expandafter\@parse@version@#2//00\@nil
193      \expandafter\@secondoftwo
194    \else
195      \expandafter\@firstoftwo
196    \fi}
197  \def\@parse@version@#1{\@parse@version@#1}
198  \end{ekernel | latexrelease}
199  \EndIncludeInRelease
200  \IncludeInRelease[0000/00/00]%
201  {\@ifl@t@r}{Guard against bad input}%
202  \def\@ifl@t@r#1#2{%
203    \ifnum\expandafter\@parse@version@#1//00\@nil<%
204      \expandafter\@parse@version@#2//00\@nil
205      \expandafter\@secondoftwo
206    \else
207      \expandafter\@firstoftwo
208    \fi}
209  \let\@parse@version@\undefined
210  \EndIncludeInRelease
211  \end{ekernel}

```

(End of definition for `\@ifl@ter`.)

```

212  \end{ekernel}
213  \end{ekernel | latexreleasefirst}
214  \def\@parse@version@#1/#2/#3#4#5\@nil{%
215  \@parse@version@dash#1-#2-#3#4\@nil
216  }

```

The `\if` test here ensures that an argument with no / or - produces 0 (actually 00).

```

217  \def\@parse@version@dash#1-#2-#3#4#5\@nil{%
218    \if\relax#2\relax\else#1\fi#2#3#4 }
219  \end{ekernel | latexreleasefirst}
220  \end{ekernel}

```

```

\@ifpackagewith \@ifpackagewith{\langle name\rangle}{\langle option-list\rangle} Checks that \langle option-list\rangle is a subset of the
  \@ifclasswith options with which \langle name\rangle was loaded.
  221 \def\@ifpackagewith{\@ifoptions\@pkgextension}
  222 \def\@ifclasswith{\@ifoptions\@clsextension}
  223 \def\@ifoptions#1#2{%
  224   \@expandtwoargs\@ifoptions{\@optionlist{#2.#1}}}

  Probably shouldn't use \CurrentOption here... (changed to \reserved@b.)
  225 \relax
  226 \if@2ekernel \else \IncludeInRelease{2017/01/01}%
  227 \else \else {\@ifoptions{Spaces in option clash check}}%
  228 \fi \if@2ekernel \else \else \fi
  229 \def\@ifoptions#1#2{%
  230   \let\reserved@a\@firstoftwo
  231   \edef\reserved@b{\zap@space#2 \empty}%
  232   \for\reserved@b:=\reserved@b\do{%
  233     \ifx\reserved@b\empty
  234     \else
  235       \expandafter\in@\expandafter{\expandafter,\reserved@b,}{, #1,}%
  236       \ifin@
  237     \else
  238       \let\reserved@a\@secondoftwo
  239     \fi
  240   \fi
  241 }%
  242 \reserved@a
  243 \if@2ekernel \else \else \fi
  244 \else \EndIncludeInRelease
  245 \else \IncludeInRelease{0000/00/00}%
  246 \else {\@ifoptions{Spaces in option clash check}}%
  247 \else \def\@ifoptions#1#2{%
  248   \let\reserved@a\@firstoftwo
  249   \for\reserved@b:=#2\do{%
  250     \ifx\reserved@b\empty
  251     \else
  252       \expandafter\in@\expandafter
  253       {\expandafter,\reserved@b,}{, #1,}%
  254     \ifin@
  255     \else
  256       \let\reserved@a\@secondoftwo
  257     \fi
  258   \fi
  259 }%
  260 \reserved@a
  261 \EndIncludeInRelease
  262 \fi

```

(End of definition for \@ifpackagewith and \@ifclasswith.)

\IfPackageLoadedTF	More public names for the commands already available for a long time.
\IfPackageLoadedWithOptionsTF	
\IfClassLoadedTF	
\IfClassLoadedWithOptionsTF	

```

  263 \relax
  264 \if@2ekernel \else \else \fi
  265 \else \IncludeInRelease{2021/11/15}%

```

```

266  \let \IfPackageLoadedTF {\IfPackageLoadedTF}{Test package loading}%
267  \let \IfClassLoadedTF {\IfClassLoadedTF}{\ifpackageloaded}
268  \let \IfPackageLoadedWithOptionsTF {\IfPackageLoadedWithOptionsTF}{\ifpackagewith}
269  \let \IfClassLoadedWithOptionsTF {\IfClassLoadedWithOptionsTF}{\ifclasswith}

```

For rollback pretend it was available since the beginning of dawn.

```

271  \let \IfPackageLoadedTF {\IfPackageLoadedTF}{Test package loading}%
272  \let \IfClassLoadedTF {\IfClassLoadedTF}{\ifpackageloaded}
273  \let \IfPackageLoadedWithOptionsTF {\IfPackageLoadedWithOptionsTF}{\ifpackagewith}
274  \let \IfClassLoadedWithOptionsTF {\IfClassLoadedWithOptionsTF}{\ifclasswith}
275  \let \EndIncludeInRelease {\EndIncludeInRelease}{}
276  \let \IfPackageLoadedTF {\IfPackageLoadedTF}{\ifpackageloaded}
277  \let \IfClassLoadedTF {\IfClassLoadedTF}{\ifclassloaded}
278  \let \IfPackageLoadedWithOptionsTF {\IfPackageLoadedWithOptionsTF}{\ifpackagewith}
279  \let \IfClassLoadedWithOptionsTF {\IfClassLoadedWithOptionsTF}{\ifclasswith}
280  \let \EndIncludeInRelease {\EndIncludeInRelease}{}
281  \let \EndIncludeInRelease {\EndIncludeInRelease}{}
282  \let \EndIncludeInRelease {\EndIncludeInRelease}{}

```

(End of definition for `\IfPackageLoadedTF` and others.)

`\ProvidesPackage` Checks that the current filename is correct, and defines `\ver@filename`.

```

283  \let \ProvidesPackage {\ProvidesPackage}{Check name with \strcmp}%
284  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currname\fi}
285  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currname\fi}
286  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currname\fi}
287  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currname\fi}
288  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currname\fi}

```

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.

```

289  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}
290  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}
291  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}
292  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}
293  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}
294  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}
295  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}
296  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}
297  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}
298  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}
299  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}
300  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}
301  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}
302  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}
303  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}
304  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}
305  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}
306  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}
307  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}
308  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}
309  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}
310  \let \ProvidesPackage {\ProvidesPackage}{\ifx\@currname\@gtempa\relax\@currpath\@currname\else\@currpath\@currname\fi}

```

```
311  {*2ekernel}
```

(End of 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.

```
312  /{2ekernel}
313  {*2ekernel | latexrelease}
314  {latexrelease}\IncludeInRelease{2020/10/01}%
315  {latexrelease}          {\@pr@videopackage}{Allow for package substitution}%
316  \def\@pr@videopackage[#1]{%
317    \expandafter\protected@xdef           %      <-- protected...
318    \csname ver@\@currname.\@currext\endcsname{#1}%
319  \expandafter\let
320    \csname ver@\@currpkg\reqd\expandafter\endcsname % Requested package
321    \csname ver@\@currname.\@currext\endcsname
322  \ifx\@currext\@clsextension
323    \typeout{Document Class: \@gtempa\space#1}%
324  \else
325    \protected@wlog{Package: \@gtempa\space#1}%
326  \fi}
```

(End of definition for \@pr@videopackage.)

- \protected@wlog This is like plain TeX's \wlog but gracefully handles protected commands.

```
327  \long\def\protected@wlog#1{\begingroup
328  \set@display@protect
329  \immediate \write \m@ne {#1}\endgroup }
```

(End of definition for \protected@wlog.)

```
330  /{2ekernel | latexrelease}
331  {latexrelease}\EndIncludeInRelease
332  {latexrelease}\IncludeInRelease{2020/02/02}%
333  {latexrelease}          {\@pr@videopackage}{Protection for package info}%
334  {latexrelease}
335  {latexrelease}\def\@pr@videopackage[#1]{%
336  {latexrelease}  \expandafter\protected@xdef           %      <-- protected...
337  {latexrelease}  \csname ver@\@currname.\@currext\endcsname{#1}%
338  {latexrelease}\ifx\@currext\@clsextension
339  {latexrelease}  \typeout{Document Class: \@gtempa\space#1}%
340  {latexrelease}  \else
341  {latexrelease}  \protected@wlog{Package: \@gtempa\space#1}%
342  {latexrelease}  \fi}
343  {latexrelease}
344  {latexrelease}\EndIncludeInRelease
345  {latexrelease}\IncludeInRelease{0000/00/00}%
346  {latexrelease}          {\@pr@videopackage}{Protection for package info}%
347  {latexrelease}
348  {latexrelease}\def\@pr@videopackage[#1]{%
349  {latexrelease}  \expandafter\xdef\csname ver@\@currname.\@currext\endcsname{#1}%
350  {latexrelease}  \ifx\@currext\@clsextension
351  {latexrelease}  \typeout{Document Class: \@gtempa\space#1}%
352  {latexrelease}  \else
353  {latexrelease}  \wlog{Package: \@gtempa\space#1}%
```

```

354 〈\latexrelease〉 \fi}
355 〈\latexrelease〉\let\protected@wlog\@undefined
356 〈\latexrelease〉
357 〈\latexrelease〉\EndIncludeInRelease
358 〈*2ekernel〉
359 \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.

```

360 〈/2ekernel〉
361 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
362 〈\latexrelease〉 {\ProvidesClass}{\Track \ProvidesPackage}%
363 〈*2ekernel | \latexrelease〉
364 \let\ProvidesClass\ProvidesPackage
365 \onlypreamble\ProvidesClass
366 〈/2ekernel | \latexrelease〉
367 〈\latexrelease〉\EndIncludeInRelease
368 〈*2ekernel〉

```

(End of 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 369 \def\ProvidesFile#1{%
370   \begingroup
371     \catcode`\ 10 %
372     \ifnum \endlinechar<256 %
373       \ifnum \endlinechar>\m@ne
374         \catcode\endlinechar 10 %
375       \fi
376     \fi
377     \makeother\%
378     \makeother\&%
379   \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}%
  \endgroup
}

```

(End of definition for `\ProvidesFile` and `\@providesfile`.)

\PassOptionsToPackage If the package has been loaded, we check that it was first loaded with the options. Otherwise we add the option list to that of the package.

```

\PassOptionsToClass 380 〈/2ekernel〉
381 〈\latexrelease〉\IncludeInRelease{2021/06/01}%
382 〈\latexrelease〉 \(\@passoptions\{Raw option lists}\}%
383 〈*2ekernel | \latexrelease〉
384 \def\@passoptions#1#2#3{%

```

```

385  \cexpl@@@filehook@set@curr@file@nNN
386  {\cexpl@@@filehook@resolve@file@subst@w #3.#1@nil}%
387  \reserved@a\reserved@b
388  \cexpl@@@filehook@clear@replacement@flag@@
389  \expandafter\protected\xdef\csname opt@\reserved@a\endcsname{%
390  \@ifundefined{opt@\reserved@a}\empty
391  {\csname opt@\reserved@a\endcsname,}%
392  \zap@space#2 \empty}%
393  \expandafter\let
394  \csname opt@#3.#1\expandafter\endcsname
395  \csname opt@\reserved@a\endcsname
Extend raw option list
396  \@ifundefined{@raw@opt@#3.#1}%
397  {\expandafter\gdef\csname @raw@opt@#3.#1\expandafter\endcsname
398  \expandafter{\#2}}%
399  {\expandafter\g@addto@macro\csname @raw@opt@#3.#1\expandafter\endcsname
400  \expandafter{\expandafter,\#2}}%
401 }
402 </2ekernel | latexrelease>
403 <| latexrelease>\EndIncludeInRelease
404 <| latexrelease>\IncludeInRelease{2020/10/01}{\@pass@options}
405 <| latexrelease> {Add file replacement in \@pass@options}%
406 <| latexrelease>
407 <| latexrelease>\def\@pass@options#1#2#3{%
408 <| latexrelease> \cexpl@@@filehook@set@curr@file@nNN
409 <| latexrelease> {\cexpl@@@filehook@resolve@file@subst@w #3.#1@nil}%
410 <| latexrelease> \reserved@a\reserved@b
411 <| latexrelease> \cexpl@@@filehook@clear@replacement@flag@@
412 <| latexrelease> \expandafter\xdef\csname opt@\reserved@a\endcsname{%
413 <| latexrelease> \@ifundefined{opt@\reserved@a}\empty
414 <| latexrelease> {\csname opt@\reserved@a\endcsname,}%
415 <| latexrelease> \zap@space#2 \empty}%
416 <| latexrelease> \expandafter\let
417 <| latexrelease> \csname opt@#3.#1\expandafter\endcsname
418 <| latexrelease> \csname opt@\reserved@a\endcsname}
419 <| latexrelease>\EndIncludeInRelease
420 <| latexrelease>\IncludeInRelease{0000/00/00}{\@pass@options}
421 <| latexrelease> {\@pass@options}%
422 <| latexrelease>
423 <| latexrelease>\def\@pass@options#1#2#3{%
424 <| latexrelease> \expandafter\xdef\csname opt@#3.#1\endcsname{%
425 <| latexrelease> \@ifundefined{opt@#3.#1}\empty
426 <| latexrelease> {\csname opt@#3.#1\endcsname,}%
427 <| latexrelease> \zap@space#2 \empty}%
428 <| latexrelease>\EndIncludeInRelease
429 <| 2ekernel>
430 \onlypreamble\@pass@options
431 \def\PassOptionsToPackage{\@pass@options\@pkgextension}
432 \def\PassOptionsToClass{\@pass@options\@clsextension}
433 \onlypreamble\PassOptionsToPackage
434 \onlypreamble\PassOptionsToClass

```

(End of definition for \PassOptionsToPackage and \PassOptionsToClass.)

\DeclareOption Adds an option as a \ds@ command, or the default \default@ds command.

\DeclareOption*

```
435 \def\DeclareOption{%
436   \let\@fileswith@pti@ns\@badrequireerror
437   \@ifstar\@defdefault@ds\@declareoption
438   \long\def\@declareoption#1#2{%
439     \xdef\@declaredoptions{\@declaredoptions,#1}%
440     \toks@{#2}%
441     \expandafter\edef\csname ds@#1\endcsname{\the\toks@}%
442   \long\def\@defdefault@ds#1{%
443     \toks@{#1}%
444     \edef\default@ds{\the\toks@}%
445     \onlypreamble\DeclareOption
446     \onlypreamble\@declareoption
447     \onlypreamble\@defdefault@ds}
```

(End of 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.

```
448 </2ekernel>
449 <latexrelease>\IncludeInRelease{2021/06/01}%
450 <latexrelease>          {\@OptionNotUsed}{filter unused option list}%
451 <*2ekernel | latexrelease>
452 \def\@remove@eq@value#1=#2\@nil{#1}

453 \def\OptionNotUsed{%
454   \ifx\@currext\@clsextension
455     \xdef\@unusedoptionlist{%
456       \ifx\@unusedoptionlist\empty\else\@unusedoptionlist,\fi
457       \expandafter\@remove@eq@value\CurrentOption=\@nil}%
458   \fi}
459 </2ekernel | latexrelease>
460 <latexrelease>\EndIncludeInRelease
461 <latexrelease>\IncludeInRelease{0000/00/00}%
462 <latexrelease>          {\@OptionNotUsed}{filter unused option list}%
463 <latexrelease>\let\@remove@eq@value\undefined

464 <latexrelease>\def\OptionNotUsed{%
465 <latexrelease>  \ifx\@currext\@clsextension
466 <latexrelease>    \xdef\@unusedoptionlist{%
467 <latexrelease>      \ifx\@unusedoptionlist\empty\else\@unusedoptionlist,\fi
468 <latexrelease>      \CurrentOption}%
469 <latexrelease>  \fi}
470 <latexrelease>\EndIncludeInRelease
471 <*2ekernel>

472 \onlypreamble\OptionNotUsed
```

(End of definition for \OptionNotUsed and \@remove@eq@value.)

\default@ds The default option code. Set by \onefilewithoptions to either \OptionNotUsed for classes, or \unknownonerror for packages. This may be reset in either case with \DeclareOption*.

```
473 % \let\default@ds\OptionNotUsed
```

(End of definition for \default@ds.)

\ProcessOptions \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.

```
474 \def\ProcessOptions{%
475   \let\ds@\empty
476   \protected@edef{\curroptions{\optionlist{\currname.\currext}}}{%
477     \@ifstar\xprocessoptions\processoptions
478     \onlypreamble\ProcessOptions
479   \def\@processoptions{%
480     \@for\CurrentOption:=\@declaredoptions\do{%
481       \ifx\CurrentOption\empty\else
482         \expandafter\in@\CurrentOption,\curroptions{%
483           ,\ifx\currext\clsextension\else\classoptionslist,\fi
484           \curroptions,}%
485       \ifin@
486         \useoption
487         \expandafter\let\csname ds@\CurrentOption\endcsname\empty
488       \fi
489     }%
490   \processoptions}
491   \onlypreamble\@processoptions
492   </2ekernel>
493   <latexrelease>\IncludeInRelease{2021/06/01}%
494   <latexrelease>           {\@xprocessoptions}{safer @xprocessoptions}%
495   <2ekernel | latexrelease>
496   \def\@xprocessoptions{%
497     \ifx\currext\clsextension\else
498       \ifx\classoptionslist\relax\else
499         \@for\CurrentOption:=\classoptionslist\do{%
500           \ifx\CurrentOption\empty\else
501             \ifundefined{ds@\detokenize\expandafter{\CurrentOption}}{}{%
502               \useoption
503               \expandafter\let\csname ds@\CurrentOption\endcsname\empty
504             }%
505           \fi}%
506         \fi
507       \fi
508     \processoptions
509   </2ekernel | latexrelease>
510   <latexrelease>\EndIncludeInRelease
511   <latexrelease>\IncludeInRelease{0000/00/00}%
512   <latexrelease>           {\@xprocessoptions}{safer @xprocessoptions}%
513   <latexrelease>\let\removeeq@value\undefined
514   <latexrelease>\def\@xprocessoptions{%
515     \ifx\currext\clsextension\else
```

```

516 <latexrelease>    \@for\CurrentOption:=\@classoptionslist\do{%
517   <latexrelease>      \ifx\CurrentOption\empty\else
518   <latexrelease>        \@expandtwoargs\in@{\, \CurrentOption,\, ,\@declaredoptions,\, }%
519   <latexrelease>        \ifin@
520   <latexrelease>          \@use@option
521   <latexrelease>          \expandafter\let\csname ds@\CurrentOption\endcsname\empty
522   <latexrelease>        \fi
523   <latexrelease>        \fi}%
524   <latexrelease>  \fi
525 <latexrelease>  \@process@pti@ns}
526 <latexrelease>\EndIncludeInRelease
527 {*2ekernel}

528 \@onlypreamble\@xprocess@ptions

```

The common part of `\ProcessOptions` and `\ProcessOptions*`.

```

529 </2ekernel>
530 {*2ekernel | latexrelease}
531 <latexrelease>\IncludeInRelease{2020/10/01}%
532 <latexrelease>          {\@process@pti@ns}{Unused options issue}%
533 \def\@process@pti@ns{%
534   \@for\CurrentOption:=\@curroptions\do{%
535     \@ifundefined{ds@\detokenize\expandafter{\CurrentOption}}{%
536       {\@use@option
537         \default@ds}}%
538       \@use@option}%

```

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`.

```

539   \@use@option}%
540
541   \let\CurrentOption\empty
542   \let\@fileswith@pti@ns\@fileswith@pti@ns
543   \AtEndOfPackage{\expandafter\let
544     \csname unprocessedoptions-\@currname.\@currext\endcsname
545     \relax}%
546   \@onlypreamble\@process@pti@ns
547   </2ekernel | latexrelease>
548   <latexrelease>\EndIncludeInRelease
549   <latexrelease>\IncludeInRelease{0000/00/00}%
550   <latexrelease>          {\@process@pti@ns}{Unused options issue}%
551   <latexrelease>
552   <latexrelease>\def\@process@pti@ns{%
553   <latexrelease>  \@for\CurrentOption:=\@curroptions\do{%
554   <latexrelease>    \@ifundefined{ds@\CurrentOption}{%
555     {\@use@option
556       \default@ds}}%
557     \@use@option}%
558   <latexrelease>  \@for\CurrentOption:=\@declaredoptions\do{%
559   <latexrelease>    \expandafter\let\csname ds@\CurrentOption\endcsname\relax}%
560   <latexrelease>  \let\CurrentOption\empty

```

```

561 <{latexrelease}> \let\@fileswith@pti@ns\@@fileswith@pti@ns
562 <{latexrelease}> \AtEndOfPackage{\let\@unprocessedoptions\relax}
563 <{latexrelease}>\EndIncludeInRelease
564 <{2ekernel}>

```

(End of definition for \ProcessOptions and \ProcessOptions*.)

\@options \@options is a synonym for \ProcessOptions* for upward compatibility with L^AT_EX2.09 style files.

```

565 \def\@options{\ProcessOptions*}
566 \onlypreamble\@options

```

(End of definition for \@options.)

\@use@option Execute the code for the current option.

```

567 <{/2ekernel}>
568 <{latexrelease}>\IncludeInRelease{2021/06/01}%
569 <{latexrelease}> {\@use@option}{filter unused option list}%
570 <{2ekernel | latexrelease}>
571 \def\@use@option{%
572   \@expandtwoargs\@removeelement
573   {\expandafter\@removeeq@value\CurrentOption=\@nil}%
574   \unusedoptionlist\unusedoptionlist
575   \csname ds@\detokenize\expandafter{\CurrentOption}\endcsname
576 <{/2ekernel | latexrelease}>
577 <{latexrelease}>\EndIncludeInRelease
578 <{latexrelease}>\IncludeInRelease{0000/00/00}%
579 <{latexrelease}> {\@use@option}{filter unused option list}%
580 <{latexrelease}>\def\@use@option{%
581   \@expandtwoargs\@removeelement\CurrentOption
582   \unusedoptionlist\unusedoptionlist
583   \csname ds@\CurrentOption\endcsname
584 <{latexrelease}>\EndIncludeInRelease
585 <{2ekernel}>
586 \onlypreamble\@use@option

```

(End of definition for \@use@option.)

\ExecuteOptions \ExecuteOptions{\{option-list\}} executes the code declared for each option.

```

587 <{/2ekernel}>
588 <{latexrelease}>\IncludeInRelease{2017/01/01}%
589 <{latexrelease}> {\ExecuteOptions}{Spaces in \ExecuteOptions}%
590 <{2ekernel | latexrelease}>
591 \def\ExecuteOptions#1{%

```

Use \@fortmp here as it is anyway cleared during \@for loop so does not change any existing names.

```

592   \edef\@fortmp{\zap@space#1 \@empty}%
593   \def\reserved@a##1\@nil{%
594     \@for\CurrentOption:=\@fortmp\do
595       {\csname ds@\CurrentOption\endcsname}%
596     \edef\CurrentOption{##1}%
597     \expandafter\reserved@a\CurrentOption\@nil}%
598 <{/2ekernel | latexrelease}>

```

```

599  \langle latexrelease\rangle\EndIncludeInRelease
600  \langle latexrelease\rangle\IncludeInRelease{0000/00/00}%
601  \langle latexrelease\rangle                                {\ExecuteOptions}{Spaces in \ExecuteOptions}%
602  \langle latexrelease\rangle\def\ExecuteOptions#1{%
603  \langle latexrelease\rangle  \def\reserved@a##1\@nil{%
604  \langle latexrelease\rangle    \@for\CurrentOption:=#1\do
605  \langle latexrelease\rangle          {\csname ds@\CurrentOption\endcsname}%
606  \langle latexrelease\rangle          \edef\CurrentOption{##1}%
607  \langle latexrelease\rangle          \expandafter\reserved@a\CurrentOption\@nil}
608  \langle latexrelease\rangle\EndIncludeInRelease
609  {*2ekernel}
610  \onlypreamble\ExecuteOptions

```

(*End of 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.

```

611  \def\documentclass{%
612    \let\documentclass\@twoclasseserror
613    \if@compatibility\else\let\usepackage\RequirePackage\fi
614    \@fileswithoptions\@clsextension}
615  \onlypreamble\documentclass

```

(*End of definition for \documentclass.*)

\documentstyle 2.09 style class ‘style’ declaration.

```

616  \def\documentstyle{%
617    \makeatletter\input{latex209.def}\makeatother
618    \documentclass}
619  \onlypreamble\documentstyle

```

(*End of definition for \documentstyle.*)

\RequirePackage Load package if not already loaded.

```

620  \def\RequirePackage{%
621    \@fileswithoptions\@pkextension}
622  \onlypreamble\RequirePackage

```

(*End of definition for \RequirePackage.*)

\LoadClass Load class.

```

623  \def\LoadClass{%
624    \ifx\@currext\@pkextension
625      \@latex@error
626        {\noexpand\LoadClass in package file}%
627        {You may only use \noexpand\LoadClass in a class file.}%
628    \fi
629    \@fileswithoptions\@clsextension}
630  \onlypreamble\LoadClass

```

(*End of 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.

```

631 〈/2ekernel〉
632 〈latexrelease〉\IncludeInRelease{2021/06/01}%
633 〈latexrelease〉                               {\@loadwithoptions}{Raw option lists load with options}%
634 〈*2ekernel | latexrelease〉
635 \def\@loadwithoptions#1#2#3{%
636   \expandafter\let\csname opt@#3.#1\expandafter\endcsname
637   \csname opt@\currname.\currname\endcsname
638   \expandafter\let\csname raw@opt@#3.#1\expandafter\endcsname
639   \csname raw@opt@\currname.\currname\endcsname
640   #2{#3}%
641 〈/2ekernel | latexrelease〉
642 〈latexrelease〉\EndIncludeInRelease
643 〈latexrelease〉\IncludeInRelease{0000/00/00}
644 〈latexrelease〉                               {\@loadwithoptions}{Raw option lists load with options}%
645 〈latexrelease〉\def\@loadwithoptions#1#2#3{%
646 〈latexrelease〉  \expandafter\let\csname opt@#3.#1\expandafter\endcsname
647 〈latexrelease〉  \csname opt@\currname.\currname\endcsname
648 〈latexrelease〉  #2{#3}%
649 〈latexrelease〉\EndIncludeInRelease
650 〈*2ekernel〉
651 \onlypreamble\@loadwithoptions

```

(End of definition for \@loadwithoptions.)

\LoadClassWithOptions Load class ‘#1’ with the current option list.

```

652 \def\LoadClassWithOptions{%
653   \@loadwithoptions{\clsextension\LoadClass}%
654 \onlypreamble\LoadClassWithOptions

```

(End of definition for \LoadClassWithOptions.)

\RequirePackageWithOptions Load package ‘#1’ with the current option list.

```

655 〈/2ekernel〉
656 〈*2ekernel | latexrelease〉
657 〈latexrelease〉\IncludeInRelease{2020/10/01}%
658 〈latexrelease〉                               {\RequirePackageWithOptions}{Unused options issue}%
659 \def\RequirePackageWithOptions{%

```

The resetting of the unprocessed options is now done on a per package basis.

```

660 \AtEndOfPackage{\expandafter\let
661   \csname unprocessedoptions-\currname.\currname\endcsname
662   \relax}%
663 \@loadwithoptions{\pkgextension\RequirePackage}%
664 \onlypreamble\RequirePackageWithOptions
665 〈/2ekernel | latexrelease〉
666 〈latexrelease〉\EndIncludeInRelease
667 〈latexrelease〉\IncludeInRelease{0000/00/00}%
668 〈latexrelease〉                               {\RequirePackageWithOptions}{Unused options issue}%
669 〈latexrelease〉
670 〈latexrelease〉\def\RequirePackageWithOptions{%
671 〈latexrelease〉  \AtEndOfPackage{\let\unprocessedoptions\relax}%

```

```

672 〈\latexrelease〉 \@loadwithoptions\@pkgextension\RequirePackage}
673 〈\latexrelease〉\EndIncludeInRelease
674 〈*2ekernel〉

```

(End of definition for `\RequirePackageWithOptions`.)

`\usepackage` To begin with, `\usepackage` produces an error. This is reset by `\documentclass`.

```

675 \def\usepackage#1{%
676   \@latex@error
677   {\noexpand \usepackage before \string\documentclass}%
678   {\noexpand \usepackage may only appear in the document
679    preamble, i.e.,\MessageBreak
680    between \noexpand\documentclass and
681    \string\begin{document}.}%
682   \@gobble}
683 \onlypreamble\usepackage

```

(End of definition for `\usepackage`.)

`\NeedsTeXFormat` Check that the document is running on the correct system.

```

684 \def\NeedsTeXFormat#1{%
685   \def\reserved@a{#1}%
686   \ifx\reserved@a\fmtname
687     \expandafter\@needsformat
688   \else
689     \@latex@error{This file needs format ‘\reserved@a’%
690                   \MessageBreak but this is ‘\fmtname’}{%
691                   The current input file will not be processed
692                   further,\MessageBreak
693                   because it was written for some other flavor of
694                   TeX.\MessageBreak\@ehd}%

```

If the file is not meant to be processed by L^AT_EX 2_ε we stop inputting it, but we do not end the run. We just end inputting the current file.

```

695   \endinput \fi}
696 \onlypreamble\NeedsTeXFormat
697 \def\@needsformat{%
698   \@ifnextchar[%
699     \@needsf@rmat
700   {}}
701 \onlypreamble\@needsformat
702 \def\@needsf@rmat[#1]{%
703   \@ifl@t@r\fmtversion{#1}{}{%
704     {\@latex@warning{no@line
705       {You have requested release ‘#1’ of LaTeX,\MessageBreak
706       but only release ‘\fmtversion’ is available}}}%
707   \onlypreamble\@needsf@rmat

```

(End of definition for `\NeedsTeXFormat`.)

`\zap@space` `\zap@space foo<space>\@empty` removes all spaces from `foo` that are not protected by `{ }` groups.

```

708 \def\zap@space#1 #2{%
709   #1%

```

```

710   \ifx#2\@empty\else\expandafter\zap@space\fi
711   #2}

```

(End of definition for \zap@space.)

\@fileswithoptions The common part of \documentclass and \usepackage.

```

712 \def\@fileswithoptions#1{%
713   \@ifnextchar[%]
714     {\@fileswithoptions#1%}
715     {\@fileswithoptions#1[]}}
716 \onlypreamble\@fileswithoptions

717 \def\@fileswithoptions#1[#2]#3{%
718   \ifnextchar[%]
719     {\@fileswithoptions#1[#2]#3%}
720     {\@fileswithoptions#1[#2]#3[]}}
721 \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.

```

722 </2ekernel>
723 <latexrelease>\IncludeInRelease{2020/10/01}%
724 <latexrelease>      {\@fileswithoptions}{\ifx tests in \@fileswithoptions}{%
725   {*2ekernel | latexrelease}
726   \def\@fileswithoptions#1[#2]#3[#4]{%
727     \ifx#1\@clsextension
728       \ifx\@classoptionslist\relax
729         \protected\@xdef\@classoptionslist{\zap@space#2 \@empty}%

```

Save raw class list.

```

730   \gdef\@raw@classoptionslist{#2}%
731   \def\reserved@a{%
732     \onefilewithoptions#3[{\#2}][{\#4}]#1%
733     \documentclasshook}%
734   \else
735     \def\reserved@a{%
736       \onefilewithoptions#3[{\#2}][{\#4}]#1%}
737   \fi
738 \else

```

build up a list of calls to \onefilewithoptions (one for each package) without thrashing the parameter stack.

```

739   \def\reserved@b##1,{%

```

If #1 is \onn nil we have reached the end of the list (older version used \nil here but \nil is undefined so \ifx equal to all undefined commands)

```

740   \ifx\@nnil##1\relax\else

```

If `\ifx\@nnil##1\@nnil` is true then #1 is (presumably) empty (Older code used `\relax` which is slightly easier to get into #1 by mistake, which would spoil this test.)

```

741      \ifx\@nnil##1\@nnil\else
742          \noexpand\@onefilewithoptions##1[{\unexpanded{#2}}][{#4}]%
743          \noexpand\@pkgextension
744          \fi
745          \expandafter\reserved@b
746          \fi}%
747          \edef\reserved@a{\zap@space#3 \empty}%
748          \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nnil,}%
749      \fi
750      \reserved@a
751 //2ekernel | latexrelease)
752 <latexrelease>\EndIncludeInRelease
753 <latexrelease>\IncludeInRelease{2017/01/01}%
754 <latexrelease> {\@fileswith@pti@ns}{ifx tests in \@fileswith@pti@ns}%
755 <latexrelease>\def\@fileswith@pti@ns#1[#2]#3[#4]{%
756 <latexrelease> \ifx#1\@clsextension
757 <latexrelease> \ifx\@classoptionslist\relax
758 <latexrelease> \xdef\@classoptionslist{\zap@space#2 \empty}%
759 <latexrelease> \def\reserved@a{%
760 <latexrelease> \@onefilewithoptions#3[{#2}][{#4}]#1%
761 <latexrelease> \@documentclasshook}%
762 <latexrelease> \else
763 <latexrelease> \def\reserved@a{%
764 <latexrelease> \@onefilewithoptions#3[{#2}][{#4}]#1}%
765 <latexrelease> \fi
766 <latexrelease> \else
767 <latexrelease> \def\reserved@b##1{%
768 <latexrelease> \ifx\@nnil##1\relax\else
769 <latexrelease> \ifx\@nnil##1\@nnil\else
770 <latexrelease> \noexpand\@onefilewithoptions##1[{\unexpanded{#2}}][{#4}]%
771 <latexrelease> \noexpand\@pkgextension
772 <latexrelease> \fi
773 <latexrelease> \expandafter\reserved@b
774 <latexrelease> \fi}%
775 <latexrelease> \edef\reserved@a{\zap@space#3 \empty}%
776 <latexrelease> \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nnil,}%
777 <latexrelease> \fi
778 <latexrelease> \reserved@a
779 <latexrelease>\EndIncludeInRelease
780 <latexrelease>\IncludeInRelease{0000/00/00}%
781 <latexrelease> {\@fileswith@pti@ns}{ifx tests in \@fileswith@pti@ns}%
782 <latexrelease>\def\@fileswith@pti@ns#1[#2]#3[#4]{%
783 <latexrelease> \ifx#1\@clsextension
784 <latexrelease> \ifx\@classoptionslist\relax
785 <latexrelease> \xdef\@classoptionslist{\zap@space#2 \empty}%
786 <latexrelease> \def\reserved@a{%
787 <latexrelease> \@onefilewithoptions#3[{#2}][{#4}]#1%
788 <latexrelease> \@documentclasshook}%
789 <latexrelease> \else
790 <latexrelease> \def\reserved@a{%
791 <latexrelease> \@onefilewithoptions#3[{#2}][{#4}]#1}%

```

```

792 <latexrelease>      \fi
793 <latexrelease>  \else
794 <latexrelease>    \def\reserved@b##1,{%
795   <latexrelease>      \ifx\@nil##1\relax\else
796   <latexrelease>        \ifx\relax##1\relax\else
797   <latexrelease>          \noexpand\@onefilewithoptions##1[##2][##4]%
798   <latexrelease>          \noexpand\@pkgextension
799   <latexrelease>        \fi
800 <latexrelease>      \expandafter\reserved@b
801 <latexrelease>    \fi}%
802 <latexrelease>    \edef\reserved@a{\zap@space#3 \empty}%
803 <latexrelease>    \edef\reserved@a{%
804 <latexrelease>      \expandafter\reserved@b\reserved@a,\@nil,}%
805 <latexrelease>  \fi
806 <latexrelease>  \reserved@a
807 <latexrelease>\EndIncludeInRelease
808 <2ekernel>
809 \onlypreamble\@fileswith@pti@ns

```

This macro is used when loading packages or classes.

Have the main argument as #1, so we only need one `\expandafter` above.

```

810 </2ekernel>
811 <*2ekernel | latexrelease>
812 <latexrelease>\IncludeInRelease{2020/10/01}%
813 <latexrelease>      {\@onefilewithoptions}{Hooks and unused options issue}%

```

Here this macro is called `\@onefilewithoptions`, but further ahead in this file it is renamed to `\load@onefilewithoptions`, and `\@onefilewithoptions` becomes a wrapper around this, used for bookkeeping when rolling back. Therefore, when in `latexrelease`, we need to define `\load@onefilewithoptions` instead, thus the extra guarded `\def` line below:

```

814 <*2ekernel>
815 \def\@onefilewithoptions#1[##2][##3]##4{%
816 </2ekernel>
817 <latexrelease>\def\load@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 substitution 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.

```

818  \expl@@filehook@set@curr@file@nNN{##1.##4}\reserved@a\reserved@b
819  \edef\reserved@c{\def\noexpand\reserved@c####1%

```

```

820      \detokenize\expandafter{\expanded{. #4}}%
821      \noexpand\@nil{\def\noexpand\reserved@a{####1}}}\reserved@c
822      \expandafter\reserved@c\reserved@a\@nil
823      \pushfilename
824      \xdef\@currname{\string\makeletter\reserved@a}%
825      \xdef\@currpath{\ifx\reserved@b\empty\else\reserved@b\fi}%
826      \global\let\@currext\@currext#4%

```

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.

```
827      \@ifl@aded\@currext\@currname
```

In the current preferred approach, a key family name will exist for processing using `\tkeys`. In that case, we replace the previous package options with the new ones, then call the key handler. Otherwise, we use the more classical clash handler.

```

828      {%
829      \@ifundefined{opt@handler@\@currname.\@currext}
830          {\@onefilewithoptions@clashchk{#2}}
831          {%
832              \expandafter\protected\edef\csname opt@\@currname.\@currext\endcsname
833                  {\zap@space#2 \empty}%
834                  \namedef{@raw@opt@\@currname.\@currext}{#2}%
835                  \nameuse{opt@handler@\@currname.\@currext}%
836          }%
837      }%
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!

```
839      \reset@ptions
```

First we take the `<name>` and `<ext>` given in the argument and check if the file exists, and issue an error otherwise asking for a correction with `\@missingfileerror`. For checking if the file exists we use `\@currpath` (usually empty) before `\@currname`.

```

840      \IfFileExists{\@currpath\@currname.\@currext}{}%
841          {\@missing@onefilewithoptions{#2}}%

```

If `\@currname` is empty (the user replied to the “Enter file name” prompt with `<RETURN>`), so stop here (do `\popfilename` to pop the item just added above).

This `\gobble` omits the date check at the end.

```

842      \ifx\@currname\empty
843          \expandafter\@gobble
844      \else

```

If the file exists, check if it was load-prevented, and otherwise do the bookkeeping with `\@filehook@file@push` then call `\set@curr@file` to set `\@curr@file` (and do any required substitution), then actually load the class/package with `\load@onefile@withoptions`. `\set@curr@file` also needs the file path.

```
845      \disable@packageload@do{\@currname.\@currext}%
```

```

846     {\@expl@@@filehook@file@push@@
847     \set@curr@file{\@currpath\@currname.\@currext}%
848     \@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.

```

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

```

Now just clean up and exit.

```

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

```

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

```

866     {\@ifl@ter\@currext{\@currname}{#3}{}}%
867     {\@latex@warning@no@line
868         {You have requested,\on@line,
869         version\MessageBreak
870         '#3' of \cls@pkg\space \@currname,\MessageBreak
871         but only version\MessageBreak
872         '\csname ver@\currname.\@currext\endcsname'\MessageBreak
873         is available}}%
874
875     \ifx\@currext\clsextension\let\LoadClass@twoloadclasserror\fi}%
876     \@popfilename
877     \@reset@options

```

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

```

877     \def\@onefilewithoptions@clashchk#1{%
878     \@if@ptions\@currext{\@currname}{#1}{}}%
879     {\@latex@error
880         {Option clash for \cls@pkg\space \@currname}%
881         {The package \@currname\space has already been loaded}

```

```

882     with options:\MessageBreak
883     \space\space[\@optionlist{\@currname.\@currext}]\MessageBreak
884     There has now been an attempt to load it
885     with options\MessageBreak
886     \space\space[#1]\MessageBreak
887     Adding the global options:\MessageBreak
888     \space\space
889     \@optionlist{\@currname.\@currext},#1\MessageBreak
890     to your \noexpand\documentclass declaration may fix this.%
891     \MessageBreak
892     Try typing \space <return> \space to proceed.}%
893     \@firstofone}

894 \let\@currpkg@reqd\@empty
895 \@onlypreamble\@onefilewithoptions
     The kernel no longer uses \@unprocessedoptions
896 \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.

```

897 \def\@missing@onefilewithoptions#1{%
898   \@missingfileerror{\@currpath\@currname}\@currext
899   \global\let\@currpath\@missingfile@area
900   \global\let\@currname\@missingfile@base
901   \global\let\@currext\@missingfile@ext}

```

Now the code that actually does the file loading:

```

\load@onefile@withoptions 902 \def\load@onefile@withoptions#1{%
903   \let\CurrentOption\@empty
904   \reset@options

```

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

```

905 \def\reserved@a{%
906   \@pass@options\@currext{#1}{\@currname}%
907   \expandafter\let
908     \csname opt@\@currpkg@reqd\expandafter\endcsname
909     \csname opt@\@currname.\@currext\endcsname
910   \expandafter\let
911     \csname @raw@opt@\@currpkg@reqd\expandafter\endcsname
912     \csname @raw@opt@\@currname.\@currext\endcsname
913   \global\expandafter
914   \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.

```

915   \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.

```

916 %-----
917   \ifx\@currext\@pkgextension

```

```

918      \UseHook{package/before}%
919      \UseOneTimeHook{package/\@currname/before}%
920  \else
921      \ifx\@currext\@clsextension
922          \UseHook{class/before}%
923          \UseOneTimeHook{class/\@currname/before}%
924      \fi
925  \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.

```

926  \InputIfFileExists{\@currpath\@currpkg@reqd}{}
927      {\@latex@error
928          {The \@cls@pkg\space\@currpkg@reqd\space failed to load}\@ehd}%
929 %-----

```

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.

```

930  \expandafter\let\csname unprocessedoptions-\@currname.\@currext\endcsname
931      \@unprocessedoptions
932  \csname\@currname.\@currext-h@k\endcsname
933  \expandafter\let\csname\@currname.\@currext-h@k\endcsname
934      \@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.

```

935  \ifx\@unprocessedoptions\relax
936      \let\@unprocessedoptions\@undefined

```

Otherwise run the per package set of unused options.

```

937  \else
938      \csname unprocessedoptions-\@currname.\@currext\endcsname
939  \fi

```

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

```

940  \expandafter\let
941      \csname unprocessedoptions-\@currname.\@currext\endcsname
942      \@undefined

```

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

```

943 %-----
944  \ifx\@currext\@pkextension
945      \UseOneTimeHook{package/\@currname/after}%
946      \UseHook{package/after}%
947  \else
948      \ifx\@currext\@clsextension
949          \UseOneTimeHook{class/\@currname/after}%
950          \UseHook{class/after}%
951      \fi
952  \fi}%
953 %-----

```

```
954 \@ifl@aded\@currname{}{\reserved@a{}}
```

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

```
955 \NewHook{package/before}
956 \NewHook{class/before}
957 \NewReversedHook{package/after}
958 \NewReversedHook{class/after}

959 </2ekernel | latexrelease>
960 <latexrelease>\EndIncludeInRelease
961 <latexrelease>\IncludeInRelease{0000/00/00}%
962 <latexrelease>      {\@onefilewithoptions}{Hooks and unused options issue}%
963 <latexrelease>
```

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

```
964 <latexrelease>\def\load@onefilewithoptions#1[#2] [#3]#4{%
965 <latexrelease>  \@pushfilename
966 <latexrelease>  \xdef@\currname{#1}%
967 <latexrelease>  \global\let\@currname\empty
968 <latexrelease>  \let\CurrentOption\empty
969 <latexrelease>  \@reset@options
970 <latexrelease>  \makeatletter
971 <latexrelease>  \def\reserved@a{%
972 <latexrelease>    \@ifl@aded\@currname{#1}%
973 <latexrelease>    {\@ifoptions\@currname{#1}{#2}{}}%
974 <latexrelease>    {\@latex@error
975 <latexrelease>      {Option clash for \@cls@pkg\space #1}%
976 <latexrelease>      {The package #1 has already been loaded
977 <latexrelease>      with options:\MessageBreak
978 <latexrelease>      \space\space[\@optionlist{#1.\@currname}]\MessageBreak
979 <latexrelease>      There has now been an attempt to load it
980 <latexrelease>      with options\MessageBreak
981 <latexrelease>      \space\space[#2]\MessageBreak
982 <latexrelease>      Adding the global options:\MessageBreak
983 <latexrelease>      \space\space
984 <latexrelease>      \@optionlist{#1.\@currname},#2\MessageBreak
985 <latexrelease>      to your \noexpand\documentclass declaration may fix this.%}
986 <latexrelease>      \MessageBreak
987 <latexrelease>      Try typing \space <return> \space to proceed.}}}%
988 <latexrelease>  {\@pass@options\@currname{#2}{#1}%
989 <latexrelease>    \global\expandafter
990 <latexrelease>    \let\csname ver@\currname.\@currname\endcsname\empty
991 <latexrelease>    \expandafter\let\csname\currname.\@currname-h@@k\endcsname\empty
992 <latexrelease>    \InputIfFileExists
993 <latexrelease>    {\currname.\@currname}%
994 <latexrelease>    {}%
995 <latexrelease>    {\@missingfileerror\currname.\@currname}%
996 <latexrelease>    \let\@unprocessedoptions\@unprocessedoptions
997 <latexrelease>    \csname\currname.\@currname-h@@k\endcsname
998 <latexrelease>    \expandafter\let\csname\currname.\@currname-h@@k\endcsname
999 <latexrelease>    \undefined
1000 <latexrelease>    \@unprocessedoptions}%
1001 <latexrelease>    \@ifl@ter\@currname{#1}{#3}{}}%
1002 <latexrelease>    {\@latex@warning@no@line}
```

```

1003 <latexrelease> {You have requested,\on@line,
1004 <latexrelease> version\MessageBreak
1005 <latexrelease> '#3' of \@cls@pkg\space #1,\MessageBreak
1006 <latexrelease> but only version\MessageBreak
1007 <latexrelease> '\csname ver@\#1.\@currext\endcsname'\MessageBreak
1008 <latexrelease> is available}}%
1009 <latexrelease> \ifx\@currext\@clsextension\let\LoadClass\@twoloadclasserror\fi
1010 <latexrelease> \@popfilename
1011 <latexrelease> \@reset@ptions}%
1012 <latexrelease> \reserved@a}
1013 <latexrelease>
1014 <latexrelease>\let \load@onefile@withoptions \@undefined
1015 <latexrelease>\let \@missing@onefilewithoptions \@undefined
1016 <latexrelease>
1017 <latexrelease>\EndIncludeInRelease
1018 <*2ekernel>

```

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

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

```

1019 \let\@@files with @ptions\@files with @ptions
1020 \@onlypreamble\@@files with @ptions

```

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

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

```

1021 \def\@reset@ptions{%
1022   \global\ifx\@currext\@clsextension
1023     \let\default@ds\OptionNotUsed
1024   \else
1025     \let\default@ds\@unknownoptionerror
1026   \fi
1027   \global\let\ds@\empty
1028   \global\let\@declaredoptions\empty
1029 \onlypreamble\@reset@ptions

```

(End of 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.

```

1030 \ifx\@begindocumenthook\@undefined
1031   \let\@begindocumenthook\empty
1032 \fi
1033 \let\@enddocumenthook\empty

```

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

\AtEndOfPackage The access functions.

```

\AtEndOfClass      1034 \def\AtEndOfPackage{%
\AtBeginDocument   1035 \expandafter\g@addto@macro\csname\@currname.\@currext-h@@k\endcsname}
\AtEndDocument    1036 \let\AtEndOfClass\AtEndOfPackage
1037 \onlypreamble\AtEndOfPackage
1038 \onlypreamble\AtEndOfClass
1039 </2ekernel>
1040 <*2ekernel | latexrelease>
1041 <latexrelease>\IncludeInRelease{2020/10/01}%
1042 <latexrelease>          {\AtBeginDocument}{Use hook system}%
1043 \DeclareRobustCommand\AtBeginDocument{\AddToHook{begindocument}}
1044 \DeclareRobustCommand\AtEndDocument {\AddToHook{enddocument}}
1045 \% \DeclareRobustCommand\AtEndDocument {\AddToHook{env/document/end}} % alternative impl
1046 </2ekernel | latexrelease>
1047 <latexrelease>\EndIncludeInRelease
1048 <latexrelease>\IncludeInRelease{0000/00/00}%
1049 <latexrelease>          {\AtBeginDocument}{Use hook system}%
1050 <latexrelease>
1051 <latexrelease>\DeclareRobustCommand\AtBeginDocument{\g@addto@macro\begin{documenthook}}
1052 <latexrelease>\DeclareRobustCommand\AtEndDocument{\g@addto@macro\end{documenthook}}
1053 <latexrelease>
1054 <latexrelease>\EndIncludeInRelease
1055 <*2ekernel>
1056 \onlypreamble\AtBeginDocument

```

(End of definition for \AtEndOfPackage and others.)

\@cls@pkg The current file type.

```

1057 \def\@cls@pkg{%
1058   \ifx\@currext\@clsextension
1059     document class%
1060   \else
1061     package%
1062   \fi}
1063 \onlypreamble\@cls@pkg

```

(End of definition for \@cls@pkg.)

\@unknownoptionerror Bad option.

```

1064 \def\@unknownoptionerror{%
1065   \@latex@error
1066   {Unknown option ‘\CurrentOption’ for \@cls@pkg\space‘\@currname’}%
1067   {The option ‘\CurrentOption’ was not declared in
1068   \@cls@pkg\space‘\@currname’, perhaps you\MessageBreak
1069   misspelled its name.
1070   Try typing \space <return>
1071   \space to proceed.}}
1072 \onlypreamble\@unknownoptionerror

```

(End of definition for \@unknownoptionerror.)

\@@unprocessedoptions Declare an error for each option, unless a \ProcessOptions occurred.

```

1073 \def\@@unprocessedoptions{%
1074   \ifx\@currname\@pkextension
1075     \protected@edef\@curroptions{\optionlist{\currname.\@currname}{}}
1076     \for\CurrentOption:=\@curroptions\do{%
1077       \ifx\CurrentOption\empty\else\unknownoptionerror\fi}%
1078     \fi}
1079 \onlypreamble\@@unprocessedoptions
1080 \onlypreamble\@@unprocessedoptions

```

(End of definition for \@@unprocessedoptions.)

\@badrequireerror \RequirePackage or \LoadClass occurs in the options section.

```

1081 \def\@badrequireerror#1[#2]#3[#4]{%
1082   \@latex@error
1083   {\noexpand\RequirePackage or \noexpand\LoadClass
1084    in Options Section}%
1085   {The \cls@pkg\space '\currname' is defective.\MessageBreak
1086    It attempts to load '#3' in the options section, i.e.,\MessageBreak
1087    between \noexpand\DeclareOption and \string\ProcessOptions.}%
1088 \onlypreamble\@badrequireerror

```

(End of definition for \@badrequireerror.)

\@twoloadclasserror Two \LoadClass in a class.

```

1089 \def\@twoloadclasserror{%
1090   \@latex@error
1091   {Two \noexpand\LoadClass commands}%
1092   {You may only use one \noexpand\LoadClass in a class file}%
1093 \onlypreamble\@twoloadclasserror

```

(End of definition for \@twoloadclasserror.)

\@twoclasseserror Two \documentclass or \documentstyle.

```

1094 \def\@twoclasseserror#1{%
1095   \@latex@error
1096   {Two \noexpand\documentclass or \noexpand\documentstyle commands}%
1097   {The document may only declare one class.}\@gobble}
1098 \onlypreamble\@twoclasseserror

```

(End of definition for \@twoclasseserror.)

4.2 Providing shipment

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

```

1099 \def\two@digits#1{\ifnum#1<10 0\fi\number#1}

```

(End of definition for \two@digits.)

\file\filecontents This environment implements inline files. The star-form does not write extra comments
\endfilecontents into the file.

```

1100 </2ekernel>
1101 <*2ekernel | latexrelease>
1102 <latexrelease>\IncludeInRelease[2020/10/01]%
1103 <latexrelease>           {\filecontents}{Define \q@curr@file directly (gh/220)}%
1104 %

```

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

```

1105 \def\filecontents{\@tempswatrue\@filesctrue
1106   \@ifnextchar[\filec@ntents@opt\filec@ntents
1107 }
1108 \@namedef{filecontents*}{\@tempswafalse\@filesctrue
1109   \@ifnextchar[\filec@ntents@opt\filec@ntents
1110 }
```

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

```

1111 \def\filec@ntents@opt[#1]{%
1112   \edef\@fortmp{\zap@space#1 \empty}%
1113   \for\reserved@a:=\@fortmp\do{%
1114     \ifcsname filec@ntents@\reserved@a\endcsname
1115       \csname filec@ntents@\reserved@a\endcsname
1116     \else
1117       \@latex@error{Unknown filecontents option \reserved@a}%
1118       {Valid options are force (or overwrite), nosearch, noheader, nowarn}%
1119     \fi}%
1120   \filec@ntents
1121 }
```

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

```

1122 \let\filec@ntents@force\@filesfalse
1123 \let\filec@ntents@overwrite\@filesfalse % alternative name
```

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

```
1124 \let\filec@ntents@noheader\@tempswafalse
```

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

```

1125 \def\filec@ntents@nosearch{%
1126   \let\filec@ntents@checkdir\@currdir
1127   \def\filec@ntents@where{in current directory}}
```

By default we search the whole tree:

```

1128 \let\filec@ntents@checkdir\empty
1129 \def\filec@ntents@where{exists on the system}
```

Option `nowarn` does not show any warning on the terminal but still writes it to the `.log`.

```

1130 \def\filec@ntents@nowarn{%
1131   \let\filec@ntents@warning\@latex@note@no@line
1132 }
```

By default we show terminal warnings.

```

1133 \let\filec@ntents@warning\@latex@warning@no@line
1134 \begingroup%
1135 \tempcnta=1
1136 \loop
1137   \catcode\tempcnta=12 %
1138   \advance\tempcnta\one %
1139 \ifnum\tempcnta<32      %
1140 \repeat      %
```

```

1141 \catcode`*=11 %
1142 \catcode`\^M\active%
1143 \catcode`\^L\active\let^\relax%
1144 \catcode`\^I\active%

1145 \gdef\filec@ntents#1{%
1146   \set@curr@file{\filec@ntents@checkdir#1}%
1147   \edef\q@curr@file{"\@curr@file"}%
```

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

```

1148 \chardef\reserved@c\ifx\directlua\@undefined 15 \else 127 \fi%
1149 \openin@\inputcheck\q@curr@file \space %
1150 \ifeof\inputcheck%
1151   \at@note@no@line%
1152     {Writing file '\@currdir\@curr@file'}%
```

```

1153   \ch@ck7\reserved@c\write\relax%
1154   \immediate\openout\reserved@c\q@curr@file\relax%
1155 \else%
```

```

1156 \if@filesw%
1157   \at@note@no@line%
1158   {File '\@curr@file' already \filec@ntents@where.\MessageBreak%
1159     Not generating it from this source}%
1160   \let\write@gobbletwo%
1161   \let\closeout@gobble%
1162 \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!

```

1163 \edef\reserved@b{\detokenize\expandafter{\jobname}}%
1164 \ifx\@curr@file\reserved@b%
1165   \if@fileswtrue%
1166 \else%
1167   \edef\reserved@b{\reserved@b\detokenize{.tex}}%
1168   \ifx\@curr@file\reserved@b
1169     \if@fileswtrue%
1170     \fi%
1171 \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.

```

1172 \ch@ck7\reserved@c\write\relax%
1173 \if@filesw% % Foul ... trying to overwrite \jobname!
1174   \at@error{Trying to overwrite '\jobname.tex'}{You can't %
1175     write to the file you are reading from!\MessageBreak%
1176     Data is written to screen instead.}%
1177 \else%
1178   \filec@ntents@warning%
1179   {Writing or overwriting file '\@curr@file'}%
1180   \immediate\openout\reserved@c\q@curr@file\relax%
1181 \fi%
1182 \fi%
1183 \fi%
```

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

```
1184  \closein\@inputcheck%
1185  \if@tempswa%
1186    \immediate\write\reserved@c{%
1187      \@percentchar\@percentchar\space%
1188      \expandafter\@gobble\string\LaTeXe file '\@curr@file'^^J%
1189      \@percentchar\@percentchar\space generated by the %
1190      '\@currenvir' \expandafter\@gobblefour\string\newenvironment^^J%
1191      \@percentchar\@percentchar\space from source '\jobname' on %
1192      \number\year/\two@digits\month/\two@digits\day.^~J%
1193      \@percentchar\@percentchar}%
1194  \fi%
1195  \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.

```
1196  \count@ 128\relax%
1197  \loop%
1198  \catcode\count@ 11\relax%
1199  \advance\count@\ @ne%
1200  \ifnum\count@<\@cclvi%
1201  \repeat%
1202 \edef\E{\@backslashchar end\string{\@currenvir\string}}%
1203 \edef\reserved@b{%
1204   \def\noexpand\reserved@b{%
1205     #####1\@####2\@####3\relax}%
1206   \reserved@b{%
1207     \ifx\relax##3\relax%
```

There was no `\end{filecontents}`

```
1208   \immediate\write\reserved@c{##1}%
1209   \else%
```

There was a `\end{filecontents}`, so stop this time.

```
1210   \edef^^M{\noexpand\end{\@currenvir}}%
1211   \ifx\relax##1\relax%
1212   \else%
```

Text before the `\end`, write it with a warning.

```
1213   \@latex@warning{Writing text '##1' before %
1214     \string\end{\@currenvir}\MessageBreak
1215     as last line of \@curr@file}%
1216   \immediate\write\reserved@c{##1}%
1217 \fi%
1218 \ifx\relax##2\relax%
1219 \else%
```

Text after the `\end`, ignore it with a warning.

```
1220   \@latex@warning{%
1221     Ignoring text '##2' after \string\end{\@currenvir}}%
1222 \fi%
1223 \fi%
1224 ^~M}%
```

```

1225  \catcode`\^^L\active%
1226  \let\L\@undefined%
1227  \def^^L{\expandafter\ifx\csname L\endcsname\relax\fi ^^J^^J}%
1228  \catcode`\^^I\active%
1229  \let\I\@undefined%
1230  \def^^I{\expandafter\ifx\csname I\endcsname\relax\fi\space}%
1231  \catcode`\^^M\active%
1232  \edef^^M##1^^M{%
1233    \noexpand\reserved@b##1\end\relax}}%
1234 \endgroup%
1235 </2ekernel | latexrelease>
1236 <latexrelease>\EndIncludeInRelease
1237 <latexrelease>\IncludeInRelease{2019/10/01}%
1238 <latexrelease> {\filec@ntents}{Spaces in file names + optional arg}%
1239 <latexrelease>
1240 <latexrelease>\def\filecontents{\tempswtrue\fileswtrue
1241 <latexrelease> \@ifnextchar[\filec@ntents@opt\filec@ntents
1242 <latexrelease>}
1243 <latexrelease>\@namedef{filecontents*}{\tempswfalse\fileswtrue
1244 <latexrelease> \@ifnextchar[\filec@ntents@opt\filec@ntents
1245 <latexrelease>}
1246 <latexrelease>\def\filec@ntents@opt[#1]{%
1247 <latexrelease> \edef\@fortmp{\zap@space#1 \empty}%
1248 <latexrelease> \for\reserved@a:=\@fortmp\do{%
1249 <latexrelease> \ifcsname filec@ntents@\reserved@a\endcsname
1250 <latexrelease> \csname filec@ntents@\reserved@a\endcsname
1251 <latexrelease> \else
1252 <latexrelease> \@late@error{Unknown filecontents option \reserved@a}%
1253 <latexrelease> {Valid options are force (or overwrite), nosearch, noheader}%
1254 <latexrelease> \fi}%
1255 <latexrelease> \filec@ntents
1256 <latexrelease>%
1257 <latexrelease>\let\filec@ntents@force\fileswfalse
1258 <latexrelease>\let\filec@ntents@overwrite\fileswfalse % alternative name
1259 <latexrelease>\let\filec@ntents@noheader\tempswfalse
1260 <latexrelease>\def\filec@ntents@nosearch{%
1261 <latexrelease> \let\filec@ntents@checkdir\currdir
1262 <latexrelease> \def\filec@ntents@where{in current directory}}
1263 <latexrelease>\let\filec@ntents@checkdir\empty
1264 <latexrelease>\def\filec@ntents@where{exists on the system}
1265 <latexrelease>\begingroup%
1266 <latexrelease>\tempcnta=1
1267 <latexrelease>\loop
1268 <latexrelease> \catcode\tempcnta=12 %
1269 <latexrelease> \advance\tempcnta\one %
1270 <latexrelease>\ifnum\tempcnta<32 %
1271 <latexrelease>\repeat %
1272 <latexrelease>\catcode`*=11 %
1273 <latexrelease>\catcode`\^^M\active%
1274 <latexrelease>\catcode`\^^L\active\let\relax\relax%
1275 <latexrelease>\catcode`\^^I\active%
1276 <latexrelease>\gdef\filec@ntents#1{%
1277 <latexrelease> \set@curr@file{\filec@ntents@checkdir#1}%
1278 <latexrelease> \edef\q@curr@file{\expandafter\quote@name\expandafter{\curr@file}}%
```

```

1279  ⟨latexrelease⟩  \chardef\reserved@c\ifx\directlua@undefined 15 \else 127 \fi%
1280  ⟨latexrelease⟩  \openin@\inputcheck\q@curr@file \space %
1281  ⟨latexrelease⟩  \ifeof@\inputcheck%
1282    \@latex@warning@no@line%
1283      {Writing file ‘\@currdir\@curr@file’}%
1284  ⟨latexrelease⟩  \ch@ck7\reserved@c\write\relax%
1285  ⟨latexrelease⟩  \immediate\openout\reserved@c\q@curr@file\relax%
1286  ⟨latexrelease⟩  \else%
1287  ⟨latexrelease⟩  \if@files w%
1288    \@latex@warning@no@line%
1289      {File ‘\@curr@file’ already \filec@ntents@where.\MessageBreak%
1290        Not generating it from this source}%
1291  ⟨latexrelease⟩  \let\write\gobbletwo%
1292  ⟨latexrelease⟩  \let\closeout\gobble%
1293  ⟨latexrelease⟩  \else%
1294  ⟨latexrelease⟩  \edef\reserved@a{\#1}%
1295  ⟨latexrelease⟩  \edef\reserved@a{\detokenize\expandafter{\reserved@a}}%
1296  ⟨latexrelease⟩  \edef\reserved@b{\detokenize\expandafter{\jobname}}%
1297  ⟨latexrelease⟩  \ifx\reserved@a\reserved@b%
1298    \@files wtrue%
1299  ⟨latexrelease⟩  \else%
1300  ⟨latexrelease⟩  \edef\reserved@b{\reserved@b\detokenize{.tex}}%
1301  ⟨latexrelease⟩  \ifx\reserved@a\reserved@b
1302    \@files wtrue%
1303  ⟨latexrelease⟩  \fi%
1304  ⟨latexrelease⟩  \fi%
1305  ⟨latexrelease⟩  \ch@ck7\reserved@c\write\relax%
1306  ⟨latexrelease⟩  \if@files w% % Foul ... trying to overwrite \jobname!
1307  ⟨latexrelease⟩  \@latex@error{Trying to overwrite ‘\jobname.tex’}{You can’t %
1308  ⟨latexrelease⟩  write to the file you are reading from!\MessageBreak%
1309  ⟨latexrelease⟩  Data is written to screen instead.}%
1310  ⟨latexrelease⟩  \else%
1311  ⟨latexrelease⟩  \@latex@warning@no@line%
1312  ⟨latexrelease⟩  {Writing or overwriting file ‘\@currdir\@curr@file’}%
1313  ⟨latexrelease⟩  \immediate\openout\reserved@c\q@curr@file\relax%
1314  ⟨latexrelease⟩  \fi%
1315  ⟨latexrelease⟩  \fi%
1316  ⟨latexrelease⟩  \fi%
1317  ⟨latexrelease⟩  \closein@\inputcheck%
1318  ⟨latexrelease⟩  \if@tempswa%
1319  ⟨latexrelease⟩  \immediate\write\reserved@c{%
1320    \percentchar\percentchar\space%
1321      \expandafter\gobble\string\LaTeXe file ‘\@curr@file’^J%
1322  ⟨latexrelease⟩  \percentchar\percentchar\space generated by the %
1323  ⟨latexrelease⟩  ‘\currenvir’ \expandafter\gobblefour\string\newenvironment^J%
1324  ⟨latexrelease⟩  \percentchar\percentchar\space from source ‘\jobname’ on %
1325  ⟨latexrelease⟩  \number\year\two@digits\month\two@digits\day.^J%
1326  ⟨latexrelease⟩  \percentchar\percentchar}%
1327  ⟨latexrelease⟩  \fi%
1328  ⟨latexrelease⟩  \let\do\makeother\dospecials%
1329  ⟨latexrelease⟩  \count@ 128\relax%
1330  ⟨latexrelease⟩  \loop%
1331  ⟨latexrelease⟩  \catcode\count@ 11\relax%
1332  ⟨latexrelease⟩  \advance\count@ \one%

```

```

1333 〈latexrelease〉      \ifnum\count@<\@cclvi%
1334 〈latexrelease〉      \repeat%
1335 〈latexrelease〉      \edef\@backslashchar{\end\string{\@currenvir\string}}%
1336 〈latexrelease〉      \edef\reserved@b{%
1337 〈latexrelease〉          \def\noexpand\reserved@b{%
1338 〈latexrelease〉              #####1\#####2\#####3\relax}%
1339 〈latexrelease〉      \reserved@b{%
1340 〈latexrelease〉          \ifx\relax##3\relax%
1341 〈latexrelease〉              \immediate\write\reserved@c{##1}%
1342 〈latexrelease〉          \else%
1343 〈latexrelease〉              \edef\@M{\noexpand\end{\@currenvir}}%
1344 〈latexrelease〉              \ifx\relax##1\relax%
1345 〈latexrelease〉                  \else%
1346 〈latexrelease〉                      \@latex@warning{Writing text ‘##1’ before %
1347 〈latexrelease〉                          \string\end{\@currenvir}\MessageBreak as last line of \@curr@file}%
1348 〈latexrelease〉                      \immediate\write\reserved@c{##1}%
1349 〈latexrelease〉                  \fi%
1350 〈latexrelease〉              \ifx\relax##2\relax%
1351 〈latexrelease〉                  \else%
1352 〈latexrelease〉                      \@latex@warning{%
1353 〈latexrelease〉                          Ignoring text ‘##2’ after \string\end{\@currenvir}}%
1354 〈latexrelease〉                  \fi%
1355 〈latexrelease〉              \fi%
1356 〈latexrelease〉              \@M}%
1357 〈latexrelease〉      \catcode`\^\active%
1358 〈latexrelease〉      \let\@L\@undefined%
1359 〈latexrelease〉      \def\@L{\expandafter\ifx\csname\@L\endcsname\relax\fi\@J\@J}%
1360 〈latexrelease〉      \catcode`\^\active%
1361 〈latexrelease〉      \let\@I\@undefined%
1362 〈latexrelease〉      \def\@I{\expandafter\ifx\csname\@I\endcsname\relax\fi\space}%
1363 〈latexrelease〉      \catcode`\^\active%
1364 〈latexrelease〉      \edef\@M{\@M\@M}%
1365 〈latexrelease〉      \noexpand\reserved@b##1\@E\@E\relax}%
1366 〈latexrelease〉\endgroup%
1367 〈latexrelease〉\EndIncludeInRelease
1368 〈latexrelease〉\IncludeInRelease{0000/00/00}%
1369 〈latexrelease〉      {\filec@ntents}{Spaces in file names + optional arg}%
1370 〈latexrelease〉
1371 〈latexrelease〉\let\filec@ntents@opt      \@undefined
1372 〈latexrelease〉\let\filec@ntents@force     \@undefined
1373 〈latexrelease〉\let\filec@ntents@overwrite  \@undefined
1374 〈latexrelease〉\let\filec@ntents@noheader   \@undefined
1375 〈latexrelease〉\let\filec@ntents@nosearch   \@undefined
1376 〈latexrelease〉\let\filec@ntents@checkdir   \@undefined
1377 〈latexrelease〉\let\filec@ntents@where     \@undefined
1378 〈latexrelease〉
1379 〈latexrelease〉\begingroup%
1380 〈latexrelease〉\@tempcnta=1
1381 〈latexrelease〉\loop
1382 〈latexrelease〉    \catcode\@tempcnta=12 %
1383 〈latexrelease〉    \advance\@tempcnta\@ne %
1384 〈latexrelease〉\ifnum\@tempcnta<32        %
1385 〈latexrelease〉\repeat
1386 〈latexrelease〉\catcode`\*=11 %

```

```

1387 〈\latexrelease〉\catcode`^\^M\active%
1388 〈\latexrelease〉\catcode`^\^L\active\let^\^L\relax%
1389 〈\latexrelease〉\catcode`^\^I\active%
1390 〈\latexrelease〉
1391 〈\latexrelease〉\gdef\filec@ntents#1{%
1392   \openin\@inputcheck#1 %
1393   \ifeof\@inputcheck%
1394     \@latex@warning@no@line%
1395     {Writing file `\'@currdir#1'}%
1396   \chardef\reserved@c15 %
1397   \ch@ck7\reserved@c\write%
1398   \immediate\openout\reserved@c#1\relax%
1399   \else%
1400     \closein\@inputcheck%
1401   \@latex@warning@no@line%
1402   {File '#1' already exists on the system.\MessageBreak%
1403   Not generating it from this source}%
1404   \let\write\@gobbletwo%
1405   \let\closeout\@gobble%
1406   \fi%
1407   \if@tempswa%
1408     \immediate\write\reserved@c{%
1409       \@percentchar\@percentchar\space%
1410       \expandafter\@gobble\string\LaTeXe file '#1'^^J%
1411       \@percentchar\@percentchar\space generated by the %
1412       '@currenvir' \expandafter\@gobblefour\string\newenvironment^^J%
1413       \@percentchar\@percentchar\space from source '\jobname' on %
1414       \number\year/\two@digits\month/\two@digits\day.^^J%
1415       \@percentchar\@percentchar}%
1416   \fi%
1417   \let\do\@makeother\dospecials%
1418   \count@ 128\relax%
1419   \loop%
1420     \catcode\count@ 11\relax%
1421     \advance\count@ \@ne%
1422     \ifnum\count@<\@cclvi%
1423     \repeat%
1424     \edef\E{\@backslashchar end\string{\@currenvir\string}}%
1425     \edef\reserved@b{%
1426       \def\noexpand\reserved@b{%
1427         #####1\#####2\#####3\relax}%
1428     \reserved@b{%
1429       \ifx\relax##3\relax%
1430         \immediate\write\reserved@c{##1}%
1431       \else%
1432         \edef^\^M{\noexpand\end{\@currenvir}}%
1433         \ifx\relax##1\relax%
1434       \else%
1435         \@latex@warning{Writing text '##1' before %
1436         \string\end{\@currenvir}\MessageBreak as last line of #1}%
1437         \immediate\write\reserved@c{##1}%
1438       \fi%
1439       \ifx\relax##2\relax%
1440     \else%

```

```

1441 〈\latexrelease〉          \@latex@warning{%
1442 〈\latexrelease〉          Ignoring text ‘##2’ after \string\end{〈\currenvir〉}%
1443 〈\latexrelease〉          \fi%
1444 〈\latexrelease〉          \fi%
1445 〈\latexrelease〉          ^~M}%
1446 〈\latexrelease〉
1447 〈\latexrelease〉  \catcode‘^~L\active%
1448 〈\latexrelease〉  \let\L\@undefined%
1449 〈\latexrelease〉  \def^~L{〈\expandafter\ifx\csname L\endcsname\relax\fi ^~J^~J}%
1450 〈\latexrelease〉  \catcode‘^~I\active%
1451 〈\latexrelease〉  \let\I\@undefined%
1452 〈\latexrelease〉  \def^~I{〈\expandafter\ifx\csname I\endcsname\relax\fi\space}%
1453 〈\latexrelease〉  \catcode‘^~M\active%
1454 〈\latexrelease〉  \edef^~M##1^~M{%
1455 〈\latexrelease〉    \noexpand\reserved@b##1\@E\@relax}%
1456 〈\latexrelease〉\endgroup%
1457 〈\latexrelease〉\EndIncludeInRelease
1458 〈*2ekernel〉

1459 \begingroup
1460 \catcode‘|=|\catcode‘\%
1461 \catcode‘\%=12
1462 \catcode‘\*=11
1463 \gdef\@percentchar{%
1464 \gdef\endfilecontents{|
1465   \immediate\closeout\reserved@c
1466   \def\T##1##2##3{|
1467     \ifx##1\@undefined\else
1468       \@latex@warning{no@line{##2 has been converted to Blank ##3e}|
1469     \fi|}
1470   \T\L{Form Feed}{Lin}||
1471   \T\I{Tab}{Spac}||
1472   \immediate\write\@unused{}}
1473 \global\let\endfilecontents*\endfilecontents

```

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

```

1474 %\onlypreamble\filecontents
1475 %\onlypreamble\endfilecontents
1476 %\onlypreamble\filecontents*
1477 %\onlypreamble\endfilecontents*
1478 \endgroup
1479 %\onlypreamble\filecontents
(End of definition for \filecontents and \endfilecontents.)

```

5 Package/class rollback mechanism

```

1480 〈/2ekernel〉
1481 〈*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.

```

1482 〈*tracerollback〉

```

```

1483 \%let\pkgcls@debug\typeout
1484 \let\pkgcls@debug\@gobble
1485 
```

(End of definition for \pkgcls@debug.)

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

```

1486 \def\requestedLaTeXdate{0}

```

(End of 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.

```

1487 \ifx\pkgcls@targetdate\@undefined
1488   \newcount\pkgcls@targetdate
1489 \fi
1490 \let\pkgcls@targetlabel\@empty
1491 \def\pkgcls@innerdate{\maxdimen}

```

(End of definition for \pkgcls@targetdate, \pkgcls@targetlabel, and \pkgcls@innerdate.)

\pkgcls@candidate \pkgcls@releasedate 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.

```

1492 \let\pkgcls@candidate\@empty
1493 \let\pkgcls@releasedate\@empty

```

(End of definition for \pkgcls@candidate and \pkgcls@releasedate.)

\load@onefilewithoptions \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.

```

1494 \ifx\load@onefilewithoptions\@undefined
1495   \let\load@onefilewithoptions\onefilewithoptions
1496 \def\onefilewithoptions#1[#2][#3]#4{%

```

First a bit of tracing normally disabled.

```
1497  {*tracerrollback}
1498  \pkgcls@debug{--- File loaded request (\noexpand\usepackage or ...)}`}
1499  \pkgcls@debug{@spaces 1: #1}`}
1500  \pkgcls@debug{@spaces 2: #2}`}
1501  \pkgcls@debug{@spaces 3: #3}`}
1502  \pkgcls@debug{@spaces 4: #4}`}
1503  (/tracerrollback)
```

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

```
1504  \def\pkgcls@name{#1}%
1505  \def\pkgcls@arg {#3}%
1506  \pkgcls@parse@date@arg{#3}%
1507  \let\pkgcls@candidate@\empty
```

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`.

```
1508  \begingroup
1509  \edef\reserved@a{%
1510  \endgroup
1511  \unexpanded{\load@onefilewithoptions#1[#2]}%
1512  [\pkgcls@mindate]%
1513  \unexpanded{#4}}%
1514  \reserved@a
1515 }
1516 \fi
```

(End of 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>
$\langle empty \rangle$	$\rightarrow \langle global-rollbackdate-as-number \rangle$	$\langle empty \rangle$	$\langle empty \rangle$
$\langle date \rangle$	$\rightarrow \langle global-rollbackdate-as-number \rangle$	$\langle empty \rangle$	$\langle date \rangle$
$=\langle date \rangle$	$\rightarrow \langle date-as-number \rangle$	$\langle empty \rangle$	$\langle empty \rangle$
$=\langle version \rangle$	$\rightarrow 1$	$\langle version \rangle$	$\langle empty \rangle$
$\langle other \rangle$	$\rightarrow \langle global-rollbackdate-as-number \rangle$	$\langle empty \rangle$	$\langle other \rangle$

where $\langle global-rollbackdate-as-number \rangle$ is a date request given via `latexrelease` or if there wasn't one 0.

```
1517 \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.

```
1518 \ifx\@nil#1\@nil
1519   \pkgcls@targetdate\requestedLaTeXdate\relax
1520   \let\pkgcls@targetlabel\@empty
1521   \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.

```
1522 \else
1523   \pkgcls@parse@date@arg@#1=\@nil\relax
1524 \fi
1525 }
```

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

```
1526 \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.

```
1527 \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.

```
1528 \ifx\@nil#1\@nil
1529   \@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 =).

```
1530 \ifnum \pkgcls@targetdate=\z@
1531   \pkgcls@targetdate\@one
1532   \def\pkgcls@innerdate{\maxdimen}%
1533   \pkgcls@parse@date@arg@version#2%
1534 \else
1535   \edef\pkgcls@innerdate{\the\pkgcls@targetdate}%
1536 \fi
1537 \let\pkgcls@mindate\@empty
1538 \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):

```

1539      \requestedLaTeXdate\relax
1540      \let\pkgcls@targetlabel\empty
1541      \def\pkgcls@innerdate{\maxdimen}%
1542      \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.

```

1543      \ifnum \pkgcls@targetdate > \z@
1544          \ifnum \parse@version#1//00@nil > \pkgcls@targetdate
1545              \@latex@warning@no@line{Suspicious rollback/min-date date given\MessageBreak
1546                  A minimal date of #1 has been specified for
1547                  \@cls@pkg\MessageBreak '\pkgcls@name'.\MessageBreak
1548                  But this is in conflict
1549                  with a rollback request to \requestedpatchdate}
1550          \fi
1551      \fi
1552  \fi
1553 }

```

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

```

1554 \def\pkgcls@parse@date@arg@version#1=%
1555 \def\pkgcls@targetlabel{#1}}

```

(End of definition for \pkgcls@parse@date@arg.)

\DeclareRelease First argument is the "name" of the release and it can be left empty if one doesn't like to give a name to the release. The second argument is that from which on this release was available (or should be used in case of minor updates). The final argument is the external file name of this release, by convention this should be *<pkg/cls-name>-<date>.(<extension>*) but this is not enforced and through this argument one can overwrite it.

```

1556 \def\DeclareRelease#1#2#3{%
1557     \ifnum\pkgcls@targetdate>\z@ % some sort of rollback request
1558     (*tracerollback)
1559         \pkgcls@debug{---\string\DeclareRelease:}%
1560         \pkgcls@debug{\@spaces 1: #1}%
1561         \pkgcls@debug{\@spaces 2: #2}%
1562         \pkgcls@debug{\@spaces 3: #3}%
1563     (/tracerollback)

```

If the date argument #2 is empty we are dealing with a special release that should be only accessible via its name; a typical use case would be a "beta" release. So if we are currently processing a date request we ignore it and otherwise we check if we can match the name and if so load the corresponding release file.

```

1564 \ifx\@nil#2\@nil
1565     \ifnum\pkgcls@targetdate=\@ne % named request
1566         \def\reserved@a{#1}%
1567         \ifx\pkgcls@targetlabel\reserved@a
1568             \pkgcls@use@this@release{#3}{}%
1569         (*tracerollback)
1570     \else

```

```

1571           \pkgcls@debug{Label doesn't match}%
1572   </tracer rollback>
1573       \fi
1574   <*tracer rollback>
1575       \else
1576           \pkgcls@debug{Date request: ignored}%
1577   </tracer rollback>
1578       \fi
1579   \else

```

If the value of \pkgcls@targetdate is greater than 1 (or in reality greater than something like 19930101) we are dealing with a rollback request to a specific date.

```
1580           \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.

```

1581           \ifnum\@parse@version#2//00\@nil
1582               >\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.

```

1583           \ifx\pkgcls@candidate\@empty
1584               \pkgcls@rollbackdate@error{#2}%
1585               \pkgcls@use@this@release{#3}{#2}%
1586           \else
1587               \pkgcls@use@this@release\pkgcls@candidate
1588                   \pkgcls@releasedate
1589           \fi
1590       \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.

```

1591           \def\pkgcls@candidate{#3}%
1592           \def\pkgcls@releasedate{#2}%
1593   <*tracer rollback>
1594       \pkgcls@debug{New candidate: #3}%
1595   </tracer rollback>
1596       \fi
1597   \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.

```

1598           \def\reserved@a{#1}%
1599           \ifx\pkgcls@targetlabel\reserved@a
1600               \pkgcls@use@this@release{#3}{#2}%
1601   <*tracer rollback>
1602       \else
1603           \pkgcls@debug{Label doesn't match}%
1604   </tracer rollback>
1605       \fi
1606   \fi

```

```
1607     \fi  
1608   \fi  
1609 }
```

(End of 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.

```
1610 \def\pkgcls@use@this@release#1#2{%
```

Before that we record the selection made inside the transcript.

```
1611 \pkgcls@show@selection{#1}{#2}{}
```

We then set the \pkgcls@targetdate to zero so that any \DeclareRelease or \DeclareCurrentRelease in the file we now load are bypassed⁴³ 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!

```
1612 \pkgcls@targetdate\z@  
1613 @@input #1\relax  
1614 \endinput  
1615 }
```

(End of 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.

```
1616 \def\pkgcls@show@selection#1#2{  
1617 <*tracer rollback>  
1618   \pkgcls@debug{Result: use #1}{%  
1619   />tracer rollback  
1620   \GenericInfo  
1621   {\@spaces\@spaces\space}{Rollback for  
1622   \@cls@pkg\space'@\currname' requested ->  
1623   \ifnum\pkgcls@targetdate>\@ne  
1624     date  
1625     \ifnum\requestedLaTeXdate=\pkgcls@targetdate  
1626       \requestedpatchdate  
1627     \else  
1628       \expandafter\gobble\pkgcls@arg  
1629     \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.

```
1630   Best approximation is  
1631   \else  
1632     version '\pkgcls@targetlabel'.\MessageBreak
```

⁴³The 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.

```

1633     This corresponds to
1634     \fi
1635     \ifx\@nil#2\@nil
1636         a special release%
1637     \else
1638         the release introduced on #2%
1639     \fi
1640     \@gobble}%
1641 }
```

(End of 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.

```

1642 \def\pkgcls@rollbackdate@error#1{%
1643   \@latex@error{Suspicious rollback date given}%
1644   {The '\@cls@pkg@space'\@currname' has no rollback data
1645   before #1 which\MessageBreak
1646   is after your requested rollback date --- so
1647   something may be wrong here.\MessageBreak
1648   Continue and we use the earliest known release.}}}
```

(End of definition for \pkgcls@rollbackdate@error.)

\DeclareCurrentRelease This declares the date (and possible name) of the current version of a package or class.

```
1649 \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).

```

1650 \ifnum\pkgcls@targetdate>\z@ % some sort of rollback request
1651 <tracerollback>
1652   \pkgcls@debug{---DeclareCurrentRelease}%
1653   \pkgcls@debug{ 1: #1}%
1654   \pkgcls@debug{ 2: #2}%
1655 </tracerollback>
```

If the value is greater than 1 we have to deal with a date request, so we parse #2 as a date and compare it with \pkgcls@targetdate.

```

1656 \ifnum\pkgcls@targetdate>\z@ % a date request
1657   \ifnum\@parse@version#2//00\@nil
1658     >\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).

```

1659   \ifx\pkgcls@candidate\@empty
1660     \pkgcls@rollbackdate@error{#2}%
1661   \else
1662     \pkgcls@use@this@release\pkgcls@candidate
1663           \pkgcls@releasedate
1664   \fi
```

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

```

1665     \else
1666         \pkgcls@show@selection{current version}{#2}%
1667     \fi
1668 \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.

```

1669 \def\reserved@af#1%
1670 \ifx\pkgcls@targetlabel\reserved@a
1671     \pkgcls@show@selection{current version}{#2}%
1672 \else
1673     \Q@latex@error{Requested version '\pkgcls@targetlabel' for
1674     '@cls@pkg\space' '@currname' is unknown}\Q@ehc
1675 \fi
1676 \fi
1677 \fi
1678 }

```

(End of 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.

```

1679 \DeclareRobustCommand\IfTargetDateBefore[1]{%
1680     \ifnum\pkgcls@innerdate <%
1681         \expandafter\@parse@version\expandafter#1//00\@nil
1682         \typeout{Exclude code introduced on #1}%
1683         \expandafter\@firstoftwo
1684     \else
1685         \typeout{Include code introduced on #1}%
1686         \expandafter\@secondoftwo
1687     \fi
1688 }

```

(End of definition for \IfTargetDateBefore.)

```
1689 </2ekernel | latexreleasefirst>
```

6 After Preamble

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

```

1690 {*afterpreamble}
1691 \NeedsTeXFormat{LaTeX2e}
1692 \ProvidesPackage{pkgindoc}
1693     [2020-08-08 v1.3m Package Interface in Document (DPC)]
1694 \def\reserved@a#1\do\@classoptionslist#2\do\filec@ntents#3\relax{%
1695     \gdef\@preamblecmds{#1#3}}

```

```
1696 \expandafter\reserved@a\@preamblecmds\relax
1697 ⟨/afterpreamble⟩
```

File V

ltkeys.dtx

1 Creating and using keyval options

As with any key–value input, using key–value pairs as package or class options has two parts: creating the key options and setting (using) them. Options created in this way *may* be used after package loading as general key–value settings: this will depend on the nature of the underlying code.

```
\DeclareKeys \DeclareKeys [{family}] {declarations}
```

Creates a series of options from a comma-separated *declarations* list. Each entry in this list is a key–value pair, with the *key* having one or more *properties*. A small number of “basic” *properties* are described below. The full range of properties, provided by *l3keys*, can also be used for more powerful processing. See *interface3* for the full details.

The basic properties provided here are

- *.code* — execute arbitrary code
- *.if* — sets a *TeX* *\if*... switch
- *.ifnot* — sets an inverted *TeX* *\if*... switch
- *.store* — stores a value in a macro
- *.usage* — defines whether the option can be given only when loading (*load*), in the preamble (*preamble*) or has no limitation on scope (*general*)

The part of the *key* before the *property* is the *name*, with the *value* working with the *property* to define the behaviour of the option.

For example, with

```
\DeclareKeys[mypkg]{  
    draft.if      = @mypkg@draft ,  
    draft.usage   = preamble ,  
    name.store    = '@mypkg@name ,  
    name.usage    = load ,  
    second-name.store = '@mypkg@other@name  
}
```

three options would be created. The option *draft* can be given anywhere in the preamble, and will set a switch called *\if@mypkg@draft*. The option *name* can only be given during package loading, and will save whatever value it is given in *\@mypkg@name*. Finally, the option *second-name* can be given anywhere, and will save its value in *\@mypkg@other@name*.

Keys created *before* the use of *\ProcessKeyOptionsact* as package options.

```
\DeclareUnknownKeyHandler \DeclareUnknownKeyHandler [family] {{code}}
```

The function `\DeclareUnknownKeyHandler` may be used to define the behavior when an undefined key is encountered. The *code* will receive the unknown key name as #1 and the value as #2. These can then be processed as appropriate, e.g. by forwarding to another package.

```
\ProcessKeyOptions \ProcessKeyOptions [family]
```

The `\ProcessKeyOptions` function is used to check the current option list against the keys defined for *family*. Global (class) options and local (package) options are checked when this function is called in a package.

```
\SetKeys \SetKeys [family] {{keyvals}}
```

Sets (applies) the explicit list of *keyvals* for the *family*: if the latter is not given, the value of `\@currname` used. This command may be used within a package to set options before or after using `\ProcessKeyOptions`.

1.1 Implementation of `ltkeys`

```
1  (@@=keys)
2  {*2ekernel}
3  \ExplSyntaxOn
```

1.2 Key properties

```
.code
.if   4  \group_begin:
.ifnot 5  \cs_set_protected:Npn \__keys_tmp:nn #1#2
.store 6  {
.usage 7  \quark_if_recursion_tail_stop:n {#1}
8  \cs_new_eq:cc
9  { \c_keys_props_root_str . #2 }
10 { \c_keys_props_root_str . #1 }
11 \__keys_tmp:nn
12 }
13 \__keys_tmp:nn
14 { code:n }           { code }
15 { legacy_if_set:n } { if }
16 { legacy_if_set_inverse:n } { ifnot }
17 { tl_set:N }         { store }
18 { usage:n }          { usage }
19 { \q_recursion_tail } { }
20 \q_recursion_stop
21 \group_end:
```

(End of definition for `.code` and others.)

1.3 Main mechanism

```
22 \cs_generate_variant:Nn \clist_put_right:Nn { Nv }
```

`\l__keys_options_clist` A single list is used for all options, into which they are collected before processing.

```
23 \clist_new:N \l__keys_options_clist
```

(End of definition for `\l_keys_options_clist`.)

`\l_keys_options_loading_bool`

Used to indicate we are in the loading phase: controls the outcome of warnings.

24 `\bool_new:N \l_keys_options_loading_bool`

`\l_keys_options:n` The main function calls functions to collect up the global and local options into `\l_keys_options_clist` before calling the underlying functions to actually do the processing. So that a suitable message is produced if the option is unknown, the special `unknown` key is set if it does not already exist for the current family, and is cleaned up afterwards if required. To allow the L^AT_EX 2_& layer to know this mechanism is active, and to deal with the key family not matching the file name, we store the family in all cases.

```
25 \cs_new_protected:Npn \l_keys_options:n #1
26   { \l_keys_options_expand_module:Nn \l_keys_options_aux:n {#1} }
27 \cs_new_protected:Npn \l_keys_options_aux:n #1
28   {
29     \cs_gset_protected:cpn { opt@handler@\currname.\current } 
30     { \ProcessKeyOptions [ #1 ] }
31     \cs_set_protected:Npn \l_keys_option_end: { }
32     \clist_clear:N \l_keys_options_clist
33     \l_keys_options_global:n {#1}
34     \l_keys_options_local:
35     \keys_if_exist:nnF {#1} { unknown }
36     {
37       \keys_define:nn {#1}
38       {
39         unknown .code:n =
40         {
41           \msg_error:nnnx { keys } { option-unknown }
42             { \l_keys_key_str } { \currname }
43         }
44       }
45       \cs_set_protected:Npn \l_keys_option_end:
46         { \keys_define:nn {#1} { unknown .undefine: } }
47     }
48   \bool_set_true:N \l_keys_options_loading_bool
49   \clist_map_variable:NNn \l_keys_options_clist \CurrentOption
50     { \keys_set:nV {#1} \CurrentOption }
51   \bool_set_false:N \l_keys_options_loading_bool
52   \AtEndOfPackage { \cs_set_eq:NN \unprocessedoptions \scan_stop: }
53   \l_keys_option_end:
54   \l_keys_options_loaded:n {#1}
55 }

56 \msg_new:nnnn { keys } { option-unknown }
57   { Unknown-option-'#1'-for-package-#2. }
58   {
59     LaTeX-has-been-asked-to-set-an-option-called-'#1'-
60     but-the-package-"\msg_module_name:n {#2}"-has-not-created-an-option-with-this-name.
61 }
```

(End of definition for `\l_keys_options:n`, `\l_keys_options_aux:n`, and `\l_keys_options_end:..`)

__keys_options_global:n Global (class) options are handled differently for L^AT_EX 2 _{ε} packages and classes. Hence this function is essentially a check on the current file type. The initial test is needed as L^AT_EX 2 _{ε} allows variables to be equal to \scan_stop:, which is usually forbidden in expl3 code.

```

62 \cs_new_protected:Npn \_\_keys_options_global:n #1
63 {
64     \cs_if_eq:NNF \@raw@classoptionslist \scan_stop:
65     {
66         \cs_if_eq:NNTF \@currext \@csextension
67         { \_\_keys_options_class:n {#1} }
68         { \_\_keys_options_package:n {#1} }
69     }
70 }
```

(End of definition for __keys_options_global:n.)

__keys_options_class:n For classes, each option (stripped of any content after =) is checked for existence as a key. If found, the option is added to the combined list for processing. On the other hand, unused options are stored up in \@unusedoptionlist. Before any of that, though, there is a simple check to see if there is an *unknown* key. If there is, then *everything* will match and the mapping can be skipped.

```

71 \cs_new_protected:Npn \_\_keys_options_class:n #1
72 {
73     \cs_if_free:cF { \@raw@opt@ \@currname . \@currext }
74     {
75         \keys_if_exist:nnTF {#1} { unknown }
76         {
77             \clist_put_right:Nv \l_\_\_keys_options_clist
78             { \@raw@opt@ \@currname . \@currext }
79         }
80     }
81     \clist_map_inline:cn { \@raw@opt@ \@currname . \@currext }
82     {
83         \exp_args:Ne \_\_keys_options_class:nnn
84         { \_\_keys_remove_equals:n {##1} }
85         {##1} {#1}
86     }
87 }
88 }
89 }
90 \cs_new_protected:Npn \_\_keys_options_class:nnn #1#2#3
91 {
92     \keys_if_exist:nnTF {#3} {#1}
93     {
94         \clist_put_right:Nn \l_\_\_keys_options_clist {#2}
95         \clist_remove_all:Nn \@unusedoptionlist {#1}
96     }
97     {
98         \clist_if_in:NnF \@unusedoptionlist {#1}
99         { \clist_put_right:Nn \@unusedoptionlist {#1} }
100    }
101 }
```

(End of definition for __keys_options_class:n and __keys_options_class:nnn.)

`_keys_options_package:n` For global options when processing a package, the tasks are slightly different from those for a class. The check is the same, but here there is nothing to do if the option is not applicable. Each valid option also needs to be removed from `\@unusedoptionlist`.

```

102 \cs_new_protected:Npn \_keys_options_package:n #1
103 {
104     \clist_map_inline:Nn \@raw@classoptionslist
105     {
106         \exp_args:Ne \_keys_options_package:nnn
107         { \_keys_remove_equals:n {##1} }
108         {##1} {#1}
109     }
110 }
111 \cs_new_protected:Npn \_keys_options_package:nnn #1#2#3
112 {
113     \keys_if_exist:nnT {#3} {#1}
114     {
115         \clist_put_right:Nn \l__keys_options_clist {#2}
116         \clist_remove_all:Nn \@unusedoptionlist {#1}
117     }
118 }
```

(End of definition for `_keys_options_package:n` and `_keys_options_package:nnn`.)

`_keys_options_local:` If local options are found, they are added to the processing list. L^AT_EX 2_E stores options for each file in a macro which may or may not exist, hence the need to use `\cs_if_exist:c`.

```

119 \cs_new_protected:Npn \_keys_options_local:
120 {
121     \cs_if_eq:NNF \@currext \@clsextension
122     {
123         \cs_if_exist:cT { \raw@opt@ \currname . \@currext }
124         {
125             \clist_put_right:Nv \l__keys_options_clist
126             { \raw@opt@ \currname . \@currext }
127         }
128     }
129 }
```

(End of definition for `_keys_options_local:.`)

`_keys_remove_equals:n` As the name suggests, this is a simple function to remove an equals sign from the input. `_keys_remove_equals:w` This is all wrapped up in an `n` function so that there will always be a sign available.

```

130 \cs_new:Npn \_keys_remove_equals:n #1
131 { \_keys_remove_equals:w #1 = \s__keys_stop }
132 \cs_new:Npn \_keys_remove_equals:w #1 = #2 \s__keys_stop { \exp_not:n {#1} }
```

(End of definition for `_keys_remove_equals:n` and `_keys_remove_equals:w`.)

1.4 The document interfaces

```
133 \cs_generate_variant:Nn \keys_define:nn { nx }
```

`_keys_options_expand_module:Nn` To deal with active characters inside the module argument whilst also expanding that argument, we use a combination of c- and f-type expansion. This works as the definitions for active UTF-8 bytes contain an `\ifincharname` test.

```

134 \cs_new_protected:Npn \__keys_options_expand_module:Nn #1#2
135   {
136     \cs:w __keys_options_expand_module:nN \use:e { \cs_end: {#2} } #1
137   }
138 \cs_new_protected:Npn \__keys_options_expand_module:nN #1#2
139   { #2 {#1} }

(End of definition for \__keys_options_expand_module:Nn and \__keys_options_expand_module:nN.)

```

\DeclareKeys Defining key options is quite straight-forward: we have an intermediate function to allow for potential set-up steps.

```

140 \NewDocumentCommand \DeclareKeys { 0 { \currname } +m }
141   { \__keys_options_expand_module:Nn \keys_define:nn {#1} {#2} }

(End of definition for \DeclareKeys. This function is documented on page 925.)

```

\DeclareUnknownKeyHandler

```

142 \NewDocumentCommand \DeclareUnknownKeyHandler { 0 { \currname } +m }
143   {
144     \cs_set_protected:cpx { __keys_unknown_handler_ #1 :nn } ##1##2 {#2}
145     \__keys_options_expand_module:Nn \keys_define:nx {#1}
146     {
147       unknown .code:n =
148         \exp_not:N \exp_args:NV
149         \exp_not:c { __keys_unknown_handler_ #1 :nn }
150         \exp_not:N \l_keys_key_str {####1}
151     }
152   }

(End of definition for \DeclareUnknownKeyHandler. This function is documented on page 926.)

```

\ProcessKeyOptions

We need to deal with the older interface from l3keys2e here: it had a mandatory argument. We can mop that up using a look-ahead, and then exploit that information to determine whether the package option handling is set up for the new approach for clash handling.

```

153 \NewDocumentCommand \ProcessKeyOptions { 0 { \currname } }
154   { \__keys_options:n {#1} }
155 \onlypreamble \ProcessKeyOptions

(End of definition for \ProcessKeyOptions. This function is documented on page 926.)

```

1.5 Option usage scope

Indicates that the load-time options for a package have been processed: once this has happened, make them unavailable either with a warning or an error.

```

156 \cs_new_protected:Npn \__keys_options_loaded:n #
157   {
158     \prop_get:NnNT \l_keys_usage_load_prop {#1} \l__keys_tmpa_tl
159     {
160       \clist_map_inline:Nn \l__keys_tmpa_tl
161       {
162         \keys_define:nn {#1}
163         {
164           ##1 .code:n =
165             \__keys_options_loaded:nn {#1} {##1}

```

```

166         }
167     }
168   }
169 }
170 \cs_new_protected:Npn \__keys_options_loaded:nn #1#2
171 {
172   \bool_if:NTF \l__keys_options_loading_bool
173     { \msg_warning:nnnn { keys } { load-option-ignored } }
174     { \msg_error:nnnn { keys } { load-only } }
175     {#1} {#2}
176 }

177 \msg_new:nnn { keys } { load-option-ignored }
178 {
179   Package~"\msg_module_name:n {#1}"~has~already~been~loaded:~
180   ignoring~load-time~option~"#2".
181 }

182 \msg_new:nnnn { keys } { load-only }
183 {
184   Key~"#2"~may~only~be~used~during~loading~of~package~
185   "\msg_module_name:n {#1}".
186 }
187 {
188   LaTeX~was~asked~to~set~a~key~called~"#2",~but~this~is~only~allowed~
189   in~the~optional~argument~when~loading~package~"\msg_module_name:n{#1}".
190 }

```

(End of definition for `__keys_options_loaded:n` and `__keys_options_loaded:nn`.)

Disable all preamble options in one shot.

```

191 \tl_gput_left:Nn \@kernel@after@begindocument
192 {
193   \prop_map_inline:Nn \l_keys_usage_preamble_prop
194   {
195     \clist_map_inline:nn {#2}
196     {
197       \keys_define:nn {#1}
198       {
199         ##1 .code:n =
200           \msg_error:nnn { keys } { preamble-only } {##1}
201       }
202     }
203   }
204 }
205 \msg_new:nnnn { keys } { preamble-only }
206 {
207   Key~"#1"~may~only~be~used~in~the~preamble.
208   LaTeX~was~asked~to~set~a~key~called~"#1",~but~this~is~only~allowed~
209   before~\begin{document}.~You~will~need~to~set~the~key~earlier.
210 }

```

1.6 General key setting

`\SetKeys` A simple wrapper.

```
211 \NewDocumentCommand \SetKeys { O { \currname } +m }
212   { \keys_options_expand_module:Nn \keys_set:nn {#1} {#2} }
213 \ExplSyntaxOff
214 
```

File W

ltfilehook.dtx

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 `\Cinput` 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^AT_EX 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 $\langle\text{file name}\rangle$ is available in `\CurrentFile`, which is then used when accessing the `file/\langle\text{file name}\rangle/before` and `file/\langle\text{file name}\rangle/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` These are:

`package/after`

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

`package/.../after`

`class/before` **`package/\langle name \rangle/before, package/\langle name \rangle/after`** These hooks are additionally called if the package name is $\langle\text{name}\rangle$ (without extension).

`class/after`

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

`class/.../after`

`class/\langle name \rangle/before, class/\langle name \rangle/after` These hooks are additionally called if the class name is $\langle\text{name}\rangle$ (without extension).

All `/after` hooks are implemented as reversed hooks.

The overall sequence of execution for `\usepackage` and friends is therefore:

```
\UseHook{\<package>/before}
\UseHook{\<package>/\<package name>/before}
  \UseHook{\<file>/before}
  \UseHook{\<file>/\<package name>.sty/before}
    <package contents>
  \UseHook{\<file>/\<package name>.sty/after}
  \UseHook{\<file>/after}

code from \AtEndOfPackage if used inside the package

\UseHook{\<package>/\<package name>/after}
\UseHook{\<package>/after}
```

and similar for class file loading, except that `package/` is replaced by `class/` and `\AtEndOfPackage` by `\AtEndOfClass`.

If a package or class is not loaded (or it was loaded before the hooks were set) none of the hooks are executed!

All class or package hooks involving the name of the class or package are implemented as one-time hooks, whereas all other such hooks are normal hooks. This allows for the following use case

```
\AddToHook{package/varioref/after}
  { ... apply my customizations if the package gets
    loaded (or was loaded already) ... }
```

without the need to first test if the package is already loaded.

1.4 Hooks for `\include` files

To manage `\include` files, L^AT_EX 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.⁴⁴ None of the hooks are executed when an `\include` file is bypassed because of an `\includeonly` declaration. They are, however, all executed if L^AT_EX makes an attempt to load the `\include` file (even if it doesn't exist and all that happens is "No file `\<filename>.tex`").

⁴⁴If 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`.

`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^AT_EX 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^AT_EX 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.⁴⁵

`include/excluded, include/<name>/excluded` The above hooks for `\include` files are only executed when the file is loaded (or more exactly the load is attempted). If, however, the `\include` file is explicitly excluded (through an `\includeonly` statement) the above hooks are bypassed and instead the `include/excluded` hook followed by the `include/<name>/excluded` hook are executed. This happens after L^AT_EX has loaded the `.aux` file for this include file, i.e., after L^AT_EX has updated its counters to pretend that the file was seen.

All `include` hooks involving the name of the included file are implemented as one-time hooks (whereas all other such hooks are normal hooks).

If you want to execute code that is run for every `\include` regardless of whether or not it is excluded, use the `cmd/include/before` or `cmd/include/after` hooks.

1.5 High-level interfaces for L^AT_EX

We do not provide any additional wrappers around the hooks (like `filehook` or `scrlfile`) 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.

⁴⁵For 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^AT_EX

```
\declare@file@substitution \declare@file@substitution {{file}} {{replacement-file}}
\undeclare@file@substitution \undeclare@file@substitution {{file}}
```

If $\langle file \rangle$ is requested for loading replace it with $\langle replacement-file \rangle$. \CurrentFile remains pointing to $\langle file \rangle$ but \CurrentFileUsed will show the file actually loaded.

The main use case for this declaration is to provide a corrected version of a package that can't be changed (due to its license) but no longer functions because of L^AT_EX 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 {{package}} {{alternate-code}}
\reenable@package@load \reenable@package@load {{package}}
```

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^AT_EX, 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 of definition for `\CurrentFile` and others. These functions are documented on page 934.)

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 `\langle path\rangle`, `\langle base\rangle`, and `\langle ext\rangle` 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 of 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 `\langle ext\rangle` (#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 of 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 (`\@onelinewithoptions`), 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 of 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:Nee #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^AT_EX's \InputIfFileExists command

Most of what we have to do is adding `\UseHook` into several L^AT_EX 2 _{ε} core commands, because of some circular dependencies in the kernel we do this only now and not in `ltfiles`.

```
\InputIfFileExists  \InputIfFileExists loads any file if it is available so we have to add the hooks
@input@file@exists@with@hooks   file/before and file/after in the right places. If the file doesn't exist no hooks
\unqu@tefilef@und   should be executed.
```

```
143  </2ekernel>
144  <latexrelease>\IncludeInRelease{2020/10/01}%
145  <latexrelease>          {\InputIfFileExists}{Hook management (files)}%
146  <*2ekernel | latexrelease>

147  \let\InputIfFileExists@\undefined
148  \DeclareRobustCommand \InputIfFileExists[2]{%
149    \IfFileExists{#1}%
150    {%
151      \@expl@@@filehook@file@push@@
152      \@filehook@set@CurrentFile
```

We pre-expand `\@filef@und` so that in case another file is loaded in the true branch of `\InputIfFileExists`, these don't change their value meanwhile. This isn't a worry with `\CurrentFile...` because they are kept in a stack.

```
153    \expandafter\@swaptwoargs\expandafter
154      {\expandafter\@input@file@exists@with@hooks
155       \expandafter{\@filef@und}}%
156      {#2}%
157      \@expl@@@filehook@file@pop@@
158    }%
159  }
160 \def\@input@file@exists@with@hooks#1{%
```

If the file exists then `\CurrentFile` holds its name. But we can't rely on that still being true after the file has been processed. Thus for using the name in the file hooks we need to preserve the name and then restore it for the `file/.../after` hook.

The hook always refers to the file requested by the user. The hook is *always* loaded for `\CurrentFile` which usually is the same as `\CurrentFileUsed`. In the case of a file replacement, the `\CurrentFileUsed` holds the actual file loaded. In any case the file names are normalized so that the hooks work on the real file name, rather than what the user typed in.

`\expl3`'s `\file_full_name:n` normalizes the file name (to factor out differences in the `.tex` extension), and then does a file lookup to take into account a possible path from `\l_file_search_path_seq` and `\input@path`. However only the file name and extension

are returned so that file hooks can refer to the file by their name only. The path to the file is returned in `\CurrentFilePath`.

```

161  \edef\reserved@a{%
162    \@expl@@@filehook@file@pop@assign@nnnn
163    {\CurrentFilePathUsed}%
164    {\CurrentFileUsed}%
165    {\CurrentFilePath}%
166    {\CurrentFile}}%
167  \expandafter\swaptwoargs\expandafter{\reserved@a}%

```

Before adding to the file list we need to make all (letter) characters catcode 11, because several packages use constructions like

```

\filename@parse{<filename>}
\ifx\filename@ext\clsextension
  ...
\fi

```

and that doesn't work if `\filename@ext` is `\detokenized`. Making `\clsextension` a string doesn't help much because some packages define their own `\<prefix>@someextension` with normal catcodes. This is not entirely correct because packages loaded (somehow) with catcode 12 alphabetic tokens (say, as the result of a `\string` or `\detokenize` command, or from a `\TeX` string like `\jobname`) will have these character tokens incorrectly turned into letter tokens. This however is rare, so we'll go for the all-letters approach (grepping the packages in `\TeX` Live didn't bring up any obvious candidate for breaking with this catcode change).

```

168  {\edef\reserved@a{\unqu@tefilef@und#1\@nil}%
169   \@addtofilelist{\string\makeletter\reserved@a}%
170   \UseHook{file/before}}%

```

The current file name is available in `\CurrentFile` so we use that in the specific hook.

```

171  \UseHook{file/\CurrentFile/before}%
172  \@@input #1% <- trailing space comes from \@filef@und
173  }%

```

And here, `\CurrentFile` is restored (by `\@expl@@@filehook@file@pop@assign@nnnn`) so we can use it once more.

```

174  \UseHook{file/\CurrentFile/after}%
175  \UseHook{file/after}%
176 \def\unqu@tefilef@und"#1" \@nil{#1}

```

Now declare the non-generic file hooks used above:

```

177 \NewHook{file/before}
178 \NewReversedHook{file/after}
179 {latexrelease}\EndIncludeInRelease
180 {/2ekernel | latexrelease}

```

Now define `\InputIfExists` 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}      {\InputIfExists}{Hook management (files)}%
183 {latexrelease}%
184 {latexrelease}\DeclareRobustCommand \InputIfExists[2]{%
185 {latexrelease}  \IfFileExists{#1}%
186 {latexrelease}  {%

```

```

187 〈\latexrelease〉 \expandafter\@swaptwoargs\expandafter
188 〈\latexrelease〉 { \@filef@und } { \#2 \addtofilelist { \#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 \addtofilelist { \#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 of 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:nnn {#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:n {#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 of definition for \__filehook_subst_add:nn and others.)

```

\use_ii_iii:n A variant of \use_... to discard the first of three arguments.

Todo: this should move to expl3

```
242 \cs_gset:Npn \use_ii_iii:n {#2 #3}
```

(End of definition for \use_ii_iii:n.)

```

243 \ExplSyntaxOff
244 ⟨/2ekernel | latexrelease⟩
245 ⟨latexrelease⟩\EndIncludeInRelease
246 ⟨*2ekernel⟩

```

For two internals we provide $\text{\LaTeX} 2_{\mathcal{E}}$ 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 of definition for \declare@file@substitution and \undeclare@file@substitution. These
functions are documented on page 937.)
```

265 ⟨@C=⟩
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.
\set@curr@file@nosearch A file name is built using \expandafter\string\csname<filename>\endcsname to avoid expanding utf8 active characters. The \csname expands the normalization machinery and the routine to resolve a file substitution, returning a control sequence with the same name as the file.

It happens that when <filename> is empty, the generated control sequence is \csname\endcsname, and doing \string on that results in the file `csnameendcsname.tex`. To guard against that we \ifx-compare the generated control sequence with the empty csname. To do so, \csname\endcsname has to be defined, otherwise it would be equal to \relax and we would have false positives. Here we define \csname\endcsname to expand to itself to avoid it matching the definition of some other control sequence.

```

267  ⟨/2ekernel⟩
268  ⟨*2ekernel | \latexrelease⟩
269  ⟨\latexrelease⟩\IncludeInRelease{2022/06/01}%
270  ⟨\latexrelease⟩           {\set@curr@file}{Setting current file name}%
271  \def\set@curr@file{%
272    \begingroup
273      \set@curr@file@aux}
274  \edef\set@curr@file@nosearch{%
275    \begingroup
276      \let\noexpand\input@path\noexpand\empty
277      \csname seq_clear:N\endcsname
278      \expandafter\noexpand\csname l_file_search_path_seq\endcsname
279      \noexpand\set@curr@file@aux}
280  \def\set@curr@file@aux#1{%
281    \escapechar\m@ne
282    \let\protect\string
283    \edef~{\string~}%
284    \expandafter\def\csname\expandafter\endcsname
285    \expandafter{\csname\endcsname}%

```

Two file names are set here: \curr@file@reqd which is the file requested by the user, and \curr@file which should be the same, except when we have a file substitution, in which case it holds the actual loaded file. \curr@file is resolved first, to check if a substitution happens. If it doesn't, \Expl@@@filehook@if@file@replaced@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.

```

286      \@expl@@@filehook@if@no@extension@nTF{#1}%
287          {\@tempswatrue}{\@tempswafalse}%
288      \@kernel@make@file@csname@\curr@file
289          \@expl@@@filehook@resolve@file@subst@w {#1}%
290      \@expl@@@filehook@if@file@replaced@@TF
291          {\@kernel@make@file@csname@\curr@file@reqd
292              \@expl@@@filehook@normalize@file@name@w{#1}%
293              \if@tempswa \@expl@@@filehook@drop@extension@N@\curr@file@reqd \fi}%
294          {\if@tempswa \@expl@@@filehook@drop@extension@N@\curr@file \fi
295              \global\let@\curr@file@reqd@\curr@file}%
296          \@expl@@@filehook@clear@replacement@flag@@
297      \endgroup}
298  (/2ekernel | latexrelease)
299  \end{InRelease}

300 \end{InRelease}\IncludeInRelease{2021/06/01}%
301 \end{InRelease}           {\set@\curr@file}{Setting current file name}%
302 \end{InRelease}\def\set@\curr@file#1{%
303 \begingroup
304 \escapechar`m@ne
305 \let\protect\string
306 \edef~{\string~}%
307 \expandafter\def\csname\expandafter\endcsname
308 \expandafter{\csname\endcsname}%
309 \end{InRelease}\@expl@@@filehook@if@no@extension@nTF{#1}%
310 \end{InRelease}          {\@tempswatrue}{\@tempswafalse}%
311 \end{InRelease}\@kernel@make@file@csname@\curr@file
312 \end{InRelease}\@expl@@@filehook@resolve@file@subst@w {#1}%
313 \end{InRelease}\@expl@@@filehook@if@file@replaced@@TF
314 \end{InRelease}          {\@kernel@make@file@csname@\curr@file@reqd
315              \@expl@@@filehook@normalize@file@name@w{#1}%
316              \if@tempswa \@expl@@@filehook@drop@extension@N@\curr@file@reqd \fi}%
317 \end{InRelease}          {\if@tempswa \@expl@@@filehook@drop@extension@N@\curr@file \fi
318              \global\let@\curr@file@reqd@\curr@file}%
319 \end{InRelease}\@expl@@@filehook@clear@replacement@flag@@
320 \endgroup}
321 \let\set@\curr@file@nosearch@\undefined
322 \end{InRelease}\End{InRelease}

323 \end{InRelease}\IncludeInRelease{2020/10/01}%
324 \end{InRelease}           {\set@\curr@file}{Setting current file name}%
325 \end{InRelease}\def\set@\curr@file#1{%
326 \begingroup
327 \escapechar`m@ne
328 \expandafter\def\csname\expandafter\endcsname
329 \expandafter{\csname\endcsname}%
330 \end{InRelease}\@expl@@@filehook@if@no@extension@nTF{#1}%
331 \end{InRelease}          {\@tempswatrue}{\@tempswafalse}%
332 \end{InRelease}\@kernel@make@file@csname@\curr@file
333 \end{InRelease}\@expl@@@filehook@resolve@file@subst@w {#1}%
334 \end{InRelease}\@expl@@@filehook@if@file@replaced@@TF
335 \end{InRelease}          {\@kernel@make@file@csname@\curr@file@reqd
336              \@expl@@@filehook@normalize@file@name@w{#1}%

```

```

337 <latexrelease>      \if@tempswa \expl@@filehook@drop@extension@N\curr@file@reqd \fi}%
338 <latexrelease>      {\if@tempswa \expl@@filehook@drop@extension@N\curr@file \fi
339 <latexrelease>      \global\let@\curr@file@reqd\curr@file}%
340 <latexrelease>      \expl@@filehook@clear@replacement@flag@@
341 <latexrelease> \endgroup
342 <latexrelease>\let\set@\curr@file@nosearch@\undefined
343 <latexrelease>\EndIncludeInRelease

344 <latexrelease>\IncludeInRelease{2019/10/01}%
345 <latexrelease>          {\set@\curr@file}{Setting current file name}%
346 <latexrelease>\def\set@\curr@file#1{%
347 <latexrelease> \begingroup
348 <latexrelease> \escapechar\m@ne
349 <latexrelease> \xdef@\curr@file{%
350 <latexrelease> \expandafter\expandafter\expandafter\unquote@name
351 <latexrelease> \expandafter\expandafter\expandafter{%
352 <latexrelease> \expandafter\string
353 <latexrelease> \csname@\firstofone#1\empty\endcsname}%
354 <latexrelease> \endgroup
355 <latexrelease> }
356 <latexrelease>\let\set@\curr@file@nosearch@\undefined
357 <latexrelease>\EndIncludeInRelease

358 <latexrelease>\IncludeInRelease{0000/00/00}%
359 <latexrelease>          {\set@\curr@file}{Setting current file name}%
360 <latexrelease>\let\set@\curr@file@\undefined
361 <latexrelease>\let\set@\curr@file@nosearch@\undefined
362 <latexrelease>\EndIncludeInRelease
363 <*2ekernel>

```

(End of definition for `\set@\curr@file` and others.)

Todo: This should get internalized using `\expl@` names

```

364 </2ekernel>
365 <*2ekernel | latexrelease>
366 <latexrelease>\IncludeInRelease{2020/10/01}%
367 <latexrelease>          {\@kernel@make@file@csname}{Make file csname}%

368 \def@\kernel@make@file@csname#1#2#3{%
369   \xdef#1{\expandafter\@set@\curr@file@aux
370     \csname\expandafter#2\@firstofone#3\@nil\endcsname}%

```

This auxiliary compares `\<filename>` with `\csname\endcsname` to check if the empty `.tex` file was requested.

```

371 \long\def@\set@\curr@file@aux#1{%
372   \expandafter\ifx\csname\endcsname#1%
373     .tex\else\string#1\fi}

```

Then we call `\expl@@filehook@set@curr@file@nNN` once for `\curr@file` to set `\CurrentFile(Path)Used` and once for `\curr@file@reqd` to set `\CurrentFile(Path)`. Here too the slower route is only used if a substitution happened, but here `\expl@@filehook@if@file@replaced@TF` can't be used because the flag is reset at the `\endgroup` above, so we check if `\curr@file` and `\curr@file@reqd` differ. This macro is issued separate from `\set@\curr@file` because it changes `\CurrentFile`, and side-effects would quickly get out of control.

```

374 \def@\filehook@set@CurrentFile{%

```

```

375  \Expl@@@filehook@set@curr@file@nNN{\curr@file}%
376    \CurrentFileUsed\CurrentFilePathUsed
377  \ifx\curr@file@reqd\curr@file
378    \let\CurrentFile\CurrentFileUsed
379    \let\CurrentFilePath\CurrentFilePathUsed
380  \else
381    \Expl@@@filehook@set@curr@file@nNN{\curr@file@reqd}%
382      \CurrentFile\CurrentFilePath
383  \fi}
384 
```

(/2ekernel | latexrelease)
 \latexrelease\EndIncludeInRelease
 (*2ekernel)

(End of definition for \filehook@set@CurrentFile , \kernel@make@file@csname , and \set@curr@file@aux.)

\@@_set_curr_file:nNN When inputting a file, `\set@curr@file` does a file lookup (in `\input@path` and `\l_file_search_path_seq`) and returns the actual file name (`\base` plus `\ext`) in `\CurrentFileUsed`, and in case there's a file substitution, the requested file in `\CurrentFile` (otherwise both are the same). Only the base and extension are returned, regardless of the input (both `path/to/file.tex` and `file.tex` end up as `file.tex` in `\CurrentFile`). The path is returned in `\CurrentFilePath`, in case it's needed.

```

387 
```

(/2ekernel | latexrelease)
 \latexrelease\IncludeInRelease[2020/10/01]%
 {\@@_set_curr_file:nNN}{Set curr file}%
 \ExplSyntaxOn
 \@@=filehook
 \cs_new_protected:Npn __filehook_set_curr_file:nNN #1
 {
 \exp_args:Nf __filehook_file_parse_full_name:nN {#1}
 __filehook_set_curr_file_assign:nnnNN
 }
 \cs_new_protected:Npn __filehook_set_curr_file_assign:nnnNN #1 #2 #3 #4 #5
 {
 \str_set:Nn #5 {#1}
 \str_set:Nn #4 {#2#3}
 }
 \ExplSyntaxOff

(/2ekernel | latexrelease)
 \latexrelease\EndIncludeInRelease
 (*2ekernel)

(End of definition for \@@_set_curr_file:nNN and \@@_set_curr_file_assign:nnnNN.)

2.7 Replacing a file and detecting loops

Start by sanitizing the file with `__filehook_file_parse_full_name:nN` then do `__filehook_file_subst_begin:nnn{\path}{\name}{\ext}`.

```

407 
```

(/2ekernel | latexrelease)
 \latexrelease\IncludeInRelease[2020/10/01]%
 {__filehook_resolve_file_subst:w}{Replace files detect loops}%
 \ExplSyntaxOn

```

412 \cs_new:Npn \__filehook_resolve_file_subst:w #1 \@nil
413   { \__filehook_file_parse_full_name:nN {#1} \__filehook_file_subst_begin:nnn }
414 \cs_new:Npn \__filehook_normalize_file_name:w #1 \@nil
415   { \__filehook_file_parse_full_name:nN {#1} \__filehook_file_name_compose:nnn }
416 \cs_new:Npn \__filehook_file_name_compose:nnn #1 #2 #3
417   { \tl_if_empty:nF {#1} { #1 / } #2#3 }

Since the file replacement is done expandably in a \csname, use a flag to remember if
a substitution happened. We use this in \set@curr@file to short-circuit some of it in
case no substitution happened (by far the most common case, so it's worth optimizing).
The flag raised during the file substitution algorithm must be explicitly cleared after the
\__filehook_if_file_replaced:TF conditional is no longer needed, otherwise further
uses of \__filehook_if_file_replaced:TF will wrongly return true.

flag_\__filehook_file_replaced
\__filehook_if_file_replaced:TF
\__filehook_clear_replacement_flag:

```

First off, start by checking if the current file ($\langle name \rangle + \langle ext \rangle$) has a declared substitution. If not, then just put that as the name (including a possible $\langle path \rangle$ in this case): this is the default case with no substitutions, so it's the first to be checked. The auxiliary __filehook_file_subst_tortoise_hare:nn sees that there's no replacement for #2#3 and does nothing else.

```

423 \cs_new:Npn \__filehook_file_subst_begin:nnn #1 #2 #3
424   {
425     \__filehook_file_subst_tortoise_hare:nn { #2#3 } { #2#3 }
426     { \__filehook_file_name_compose:nnn {#1} {#2} {#3} }
427   }
428 \ExplSyntaxOff
429 {/2ekernel | latexrelease}
430 {latexrelease}\EndIncludeInRelease
431 {*2ekernel}

```

2.7.1 The Tortoise and Hare algorithm

If there is a substitution ($\langle true \rangle$ in the first \cs_if_exist:cTF below), then first check if there is no substitution down the line: this should be the second most common case, of one file replaced by another. In that case just leave the substitution there and the job is done. If any substitution happens, then the \flag___filehook_file_replaced is raised (conditionally, because checking if a flag is raised is much faster than raising it over and over again).

If, however there are more substitutions, then we need to check for a possible loop in the substitutions, which would otherwise put TeX in an infinite loop if just an exhaustive expansion was used.

To detect a loop, the *Tortoise and Hare* algorithm is used. The name of the algorithm is an analogy to Aesop's fable, in which the Hare outruns a Tortoise. The two pointers here are the csnames which contains each file replacement, both of which start at the position zero, which is the file requested. In the inner part of the macro below, __filehook_file_subst_loop:cc is called with \@file-subst@ $\langle file \rangle$ and \@file-subst@\@file-subst@ $\langle file \rangle$; that is, the substitution of $\langle file \rangle$ and the substitution of that substitution: the Tortoise walks one step while the Hare walks two.

Within `__filehook_file_subst_loop:NN` the two substitutions are compared, and if they lead to the same file it means that there is a loop in the substitutions. If there's no loop, `__filehook_file_subst_tortoise_hare:nn` is called again with the Tortoise at position 1 and the hare at 2. Again, the substitutions are checked ahead of the Hare pointer to check that it won't run too far; in case there is no loop in the declarations, eventually one of the `\cs_if_exist:cTF` below will go `<false>` and the algorithm will end; otherwise it will run until the Hare reaches the same spot as the tortoise and a loop is detected.

```

432 〈/2ekernel〉
433 〈*2ekernel | latexrelease〉
434 〈latexrelease〉\IncludeInRelease{2020/10/01}%
435 〈latexrelease〉 {\__filehook_file_subst_tortoise_hare:nn}{Tortoise and Hare}%
436 \ExplSyntaxOn
437 \cs_new:Npn \__filehook_file_subst_tortoise_hare:nn #1 #2 #3
438 {
439     \cs_if_exist:cTF { @file-subst@ #2 }
440     {
441         \flag_if_raised:nF { __filehook_file_replaced }
442         { \flag_raise:n { __filehook_file_replaced } }
443         \cs_if_exist:cTF { @file-subst@ \use:c { @file-subst@ #2 } }
444         {
445             \__filehook_file_subst_loop:cc
446             { @file-subst@ #1 }
447             { @file-subst@ \use:c { @file-subst@ #2 } }
448         }
449         { \use:c { @file-subst@ #2 } }
450     }
451     { #3 }
452 }
```

This is just an auxiliary to check if a loop was found, and continue the algorithm otherwise. If a loop is found, the `.tex` file is used as fallback and `__filehook_file_subst_cycle_error:cN` is called to report the error.

```

453 \cs_new:Npn \__filehook_file_subst_loop:NN #1 #2
454 {
455     \token_if_eq_meaning:NNTF #1 #2
456     {
457         .tex
458         \__filehook_file_subst_cycle_error:cN { @file-subst@ #1 } #1
459     }
460     { \__filehook_file_subst_tortoise_hare:nn {#1} {#2} {#2} }
461 }
462 \cs_generate_variant:Nn \__filehook_file_subst_loop:NN { cc }
```

Showing this type of error expandably is tricky, as we have a very limited amount of characters to show and a potentially large list. As a work around, several errors are printed, each showing one step of the loop, until all the error messages combined show the loop.

```

463 \cs_new:Npn \__filehook_file_subst_cycle_error:NN #1 #2
464 {
465     \msg_expandable_error:nnff { latex2e } { file-cycle }
466     {#1} { \use:c { @file-subst@ #1 } }
467     \token_if_eq_meaning:NNF #1 #2
```

```

468      { \__filehook_file_subst_cycle_error:cN { @file-subst@ #1 } #2 }
469    }
470 \cs_generate_variant:Nn \__filehook_file_subst_cycle_error:NN { c }

And the error message:

471 \msg_new:nnn { latex2e } { file-cycle }
472   { File-loop!~#1-replaced~by~#2... }

(End of definition for \__filehook_resolve_file_subst:w and others.)

473 \ExplSyntaxOff
474 </2ekernel | latexrelease>
475 <latexrelease>\EndIncludeInRelease
476 <*2ekernel>
477 <@C=›

```

2.8 Preventing a package from loading

We support the use case of preventing a package from loading but not any other type of files (e.g., classes).

```

\disable@package@load \disable@package@load defines \c@pkg-disable@{package} to expand to some code #2
\reenable@package@load instead of loading the package.

\@disable@packageload@do
478 </2ekernel>
479 <*2ekernel | latexrelease>
480 <latexrelease>\IncludeInRelease{2020/10/01}%
481 <latexrelease>          {\c@disable@package@load}{Disable packages}%
482 \def\c@enable@package@load#1#2{%
483   \global\c@nameuse{\c@pkg-disable@#1.\c@pkgtension}{#2}}

```

Here we check if a control sequence named \c@pkg-disable@{name}.sty is defined, and if so don't use the package loading code #2, but use the replacement code stored in that control sequence, write something to the log, and then prevent \c@onefilewithoptions from sanity-checking the requested package date (the \expandafter here triggers one in \c@onefilewithoptions that ends a conditional there, and the \c@gobbletwo removes the date checking code from the input stream).

```

484 \def\c@enable@packageload@do#1#2{%
485   \c@ifundefined{\c@pkg-disable@#1}%
486     {#2}%
487     {\c@nameuse{\c@pkg-disable@#1}{%
488       \c@latex@info{Package '#1' has been disabled.%}
489       \MessageBreak Load request ignored}%
490     \expandafter\c@gobbletwo}}}

\reenable@package@load undefines \c@pkg-disable@{package} to reallow loading
a package.

491 \def\c@enable@package@load#1{%
492   \global\expandafter\let
493     \c@csname \c@pkg-disable@#1.\c@pkgtension \endcsname \c@undefined}%
494 </2ekernel | latexrelease>
495 <latexrelease>\EndIncludeInRelease
496 <latexrelease>\IncludeInRelease{0000/00/00}%
497 <latexrelease>          {\c@enable@package@load}{Disable packages}%
498 <latexrelease>

```

```

499 〈\latexrelease〉\let\disable@package@load  \@undefined
500 〈\latexrelease〉\let\@disable@packageload@do \@undefined
501 〈\latexrelease〉\let\reenable@package@load  \@undefined
502 〈\latexrelease〉\EndIncludeInRelease
503 〈*2ekernel〉

(End of definition for \disable@package@load, \reenable@package@load, and
 \@disable@packageload@do. These functions are documented on page 937.)
```

2.9 High-level interfaces for L^AT_EX

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^AT_EX 2 _{ε} names to allow for internal commands to be used outside this module (and in parts that still use L^AT_EX 2 _{ε} syntax). We have to unset the @@ since we want double “at” sign in place of double underscores.

```

504 〈@@=〉
505 〈/2ekernel〉
506 〈*2ekernel | latexrelease〉
507 〈\latexrelease〉\IncludeInRelease{2020/10/01}%
508 〈\latexrelease〉    {\@expl@@@filehook@if@no@extension@nTF}{2e tmp interfaces}%
509 \ExplSyntaxOn

510 \cs_new_eq:NN \@expl@@@filehook@if@no@extension@nTF
511           \__filehook_if_no_extension:nTF
512 \cs_new_eq:NN \@expl@@@filehook@set@curr@file@nNN
513           \__filehook_set_curr_file:nNN
514 \cs_new_eq:NN \@expl@@@filehook@resolve@file@subst@w
515           \__filehook_resolve_file_subst:w
516 \cs_new_eq:NN \@expl@@@filehook@normalize@file@name@w
517           \__filehook_normalize_file_name:w
518 \cs_new_eq:NN \@expl@@@filehook@if@file@replaced@TF
519           \__filehook_if_file_replaced:TF
520 \cs_new_eq:NN \@expl@@@filehook@clear@replacement@flag@%
521           \__filehook_clear_replacement_flag:
522 \cs_new_eq:NN \@expl@@@filehook@drop@extension@N
523           \__filehook_drop_extension:N
524 \cs_new_eq:NN \@expl@@@filehook@file@push@%
525           \__filehook_file_push:
526 \cs_new_eq:NN \@expl@@@filehook@file@pop@%
527           \__filehook_file_pop:
528 \cs_new_eq:NN \@expl@@@filehook@file@pop@assign@nnnn
529           \__filehook_file_pop_assign:nnnn
530 \ExplSyntaxOff
```

This one specifically has to be undefined because it is left over in the input stream from `\InputIfFileExists` and executed when `\textrun` 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 `\textrun`.

```

531 </2ekernel | \textrun>
532 <\textrun>\EndIncludeInRelease
533 <\textrun>
534 <\textrun>\IncludeInRelease{0000/00/00}%
535 <\textrun>    {\@expl@@filehook@if@no@extension@nTF}{2e tmp interfaces}%
536 <\textrun>\let\@expl@@filehook@file@pop@@\relax
537 <\textrun>\EndIncludeInRelease
538 <*2ekernel>

This ends the kernel code in this file.

539 </2ekernel>
```

3 A sample package for structuring the log output

```

540 <*structuredlog>
541 <@=filehook>
542 \ProvidesExplPackage
543     {structuredlog}{\ltfilehookdate}{\ltfilehookversion}
544     {Structuring the TeX transcript file}

\g_filehook_nesting_level_int
Stores the current package nesting level.

545 \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.

546 \int_gset:Nn \g_filehook_nesting_level_int
547   { ( \tl_count:N \currnamestack ) / 3 - 1 }

(End of 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.

548 \cs_new_protected:Npn \__filehook_log_file_record:n #1
549   {
550     \str_if_eq:nnT {#1} {START} { \int_gincr:N \g_filehook_nesting_level_int }
551     \iow_term:x
552     {
553       \prg_replicate:nn { \g_filehook_nesting_level_int } { = } ~
554       ( LEVEL ~ \int_use:N \g_filehook_nesting_level_int \c_space_tl #1 ) ~
555       \CurrentFileUsed

If there was a file replacement, show that as well:

556       \str_if_eq:NNF \CurrentFileUsed \CurrentFile
557         { ~ ( \CurrentFile \c_space_tl requested ) }
558         \iow_newline:
559       }
560     \str_if_eq:nnT {#1} {STOP} { \int_gdecr:N \g_filehook_nesting_level_int }
561   }
```

Now just hook the macro above in the generic `file/before...`

```

562 \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.
563 \AddToHookNext{file/after}
564   { \AddToHook{file/after}{ \_filehook_log_file_record:n { STOP } } }
(End of definition for \_filehook_log_file_record:n.)
565 <@@=
566 </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:

```

567 <*2ekernel>
568 \declare@file@substitution{atveryend.sty}{atveryend-ltx.sty}
569 </2ekernel>

```

Here is the package file we point to:

```

570 <*atveryend-ltx>
571 \ProvidesPackage{atveryend-ltx}
572 [2020/08/19 v1.0a
573   Emulation of the original atveryend package^^Jwith kernel methods]

```

Here are new definitions for its interfaces now pointing to the hooks in `\enddocument`

```

574 \newcommand\AfterLastShipout {\AddToHook{\enddocument/afterlastpage}}
575 \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`.

```

576 \newcommand\AtEndAfterFileList{\AddToHook{\enddocument/info}}
577 \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.

```

578 \ExplSyntaxOn
579 \newcommand\BeforeClearDocument[1]
580   { \AtEndDocument{#1}
581     \atveryend@DEPRECATED{BeforeClearDocument \tl_to_str:n{#1}}
582   }
583 \cs_new:Npn\atveryend@DEPRECATED #1
584   {\iow_term:x{=====~DEPRECATED~USAGE~#1~=====}}
585 \ExplSyntaxOff
(End of definition for \BeforeClearDocument.)
586 </atveryend-ltx>

```

File X

ltshipout.dtx

1 Introduction

The code provides an interface to the `\shipout` primitive of T_EX 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.⁴⁶

1.1 Overloading the `\shipout` primitive

`\shipout` With this implementation T_EX’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^AT_EX 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.

⁴⁶Heiko’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^AT_EX 2_E names).⁴⁷ 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.

⁴⁷Might 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 This hook is executed after foreground and/or background material has been added, i.e., just in front of the actual shipout operation. Its purpose is to allow manipulation of the finalized box (stored in `\ShipoutBox`) with the extra material also in place (which is not yet the case in `shipout/before`).

It cannot be used to cancel the shipout operation via `\DiscardShipoutBox` (that has to happen in `shipout/before`, if desired!)

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.⁴⁸

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` and `shipout/after`) are added inside `hboxes` to the box being shipped out in the following order:

⁴⁸In `LATEX 2ε` that was already existing, but implemented using a box register with the name `\@begindvibox`.

<code>shipout/firstpage</code>	only on the first page
<code>shipout/background</code>	
<code><boxed content of \ShipoutBox></code>	
<code>shipout/foreground</code>	
<code>shipout/lastpage</code>	only on the last page

If any of the hooks has no code then the corresponding box is added at that point.

Once the (page) box has got the above extra content it can again be manipulated using the `shipout` hook and then is shipped out for real.

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^AT_EX commands

`\AtBeginDvi` `\AtBeginDvi {<code>}`
`\AtEndDvi`

`\AtBeginDvi` is the existing L^AT_EX 2_E interface to fill the `shipout/firstpage` hook. This is not really a good name as it is not just supporting `.dvi` but also `.pdf` output or `.xdv`.

`\AtEndDvi` is the counterpart that was not available in the kernel but only through the package `atenddvi`. It fills the `shipout/lastpage` hook.

Neither interface can set a code label but uses the current default label.

As these two wrappers have been available for a long time we continue offering them (but not enhancing them, e.g., by providing support for code labels).

For new code we strongly suggest using the high-level hook management commands directly instead of “randomly-named” wrappers. This will lead to code that is easier to understand and to maintain and it also allows you to set code labels if needed.

For this reason we do not provide any other “new” wrapper commands for the above hooks in the kernel, but only keep the existing ones for backward compatibility.

1.4 Special commands for use inside the hooks

```
\DiscardShipoutBox \AddToHookNext {shipout/before} {...\DiscardShipoutBox...}
```

\shipout_discard:

The `\DiscardShipoutBox` declaration (L3 name `\shipout_discard:`) requests that on the next shipout the page box is thrown away instead of being shipped to the `.dvi` or `.pdf` file.

Typical applications wouldn't do this unconditionally, but have some processing logic that decides to use or not to use the page.

Note that if this declaration is used directly in the document it may depend on the placement to which page it applies, given that L^AT_EX 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 — not done (yet)

`\DiscardShipoutBox` cannot be used in any of the `shipout/...` hooks other than `shipout/before`.

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`, `shipout/background` or `shipout/foreground`.⁴⁹ If `atbegshi` gets loaded then those commands become available as public functions with their original names as given below.

1.5 Provided L^AT_EX callbacks

pre_shipout_filter Under L^AT_EX the `pre_shipout_filter` Lua callback is provided which gets called directly after the `shipout` hook, 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`.

⁴⁹If that assumption turns out to be wrong it would be trivial to change them to public functions (right now they are private).

1.6 Information counters

```
\ ReadonlyShipoutCounter \ifnum\ ReadonlyShipoutCounter=...
\g_shipout_READONLY_int \int_use:N \g_shipout_READONLY_int % expl3 usage
```

This integer holds the number of pages shipped out up to now (including the one to be shipped out when inside the output routine). More precisely, it is incremented only after it is clear that a page will be shipped out, i.e., after the `shipout/before` hook (because that might discard the page)! In contrast `shipout/after` sees the incremented value.

Just like with the `page` counter its value is only accurate within the output routine. In the body of the document it may be off by one as the output routine is called asynchronously!

Also important: it *must not* be set, only read. There are no provisions to prevent that restriction, but if you manipulate it, chaos will be the result. To emphasize this fact it is not provided as a L^AT_EX counter but as a T_EX counter (i.e., a command), so `\Alph{\ReadonlyShipoutCounter}` etc, would not work.

```
totalpages \arabic{totalpages}
\g_shipout_totalpages_int \int_use:N \g_shipout_totalpage_int % expl3 usage
```

In contrast to `\ReadonlyShipoutCounter`, the `totalpages` counter is a L^AT_EX 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 L^AT_EX. 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 L^AT_EX 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 \DebugShipoutsOn
\DebugShipoutsOff \DebugShipoutsOff
\shipout_debug_on: \shipout_debug_on:
\shipout_debug_off: \shipout_debug_off:
```

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^AT_EX 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^AT_EX 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^AT_EX mechanisms and updates, for example, the `_READONLYSHIPOUTCOUNTER` counter. Please use `\RawShipout` for new code if you want to bypass the before, foreground and background hooks.

\AtBeginShipoutInit By default `atbegshi` delayed its action until `\begin{document}`. This command was forcing it in an earlier place. With the new concept it does nothing.

\AtBeginShipout `\AtBeginShipout{\<code>} ≡ \AddToHook{shipout/before}{\<code>}`
\AtBeginShipoutNext `\AtBeginShipoutNext{\<code>} ≡ \AddToHookNext{shipout/before}{\<code>}`

This is equivalent to filling the `shipout/before` hook by either using `\AddToHook` or `\AddToHookNext`, respectively.

\AtBeginShipoutFirst The `atbegshi` names for `\AtBeginDvi` and `\DiscardShipoutBox`.
\AtBeginShipoutDiscard

2.2 Emulating `everyshi`

The `everyshi` package is providing commands to run arbitrary code just before the shipout starts. One point of difference: in the new shipout hooks the page is available as `\ShipoutBox` for inspection of change, one should not manipulate box 255 directly inside `shipout/before`, so old code doing this would change to use `\ShipoutBox` instead of 255 or `\@ccly`.

\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{\<code>} ≡ \AddToHook{shipout/background}{\put(1in,-1in){\<code>}}`

`\AddEverypageHook` is adding something into the background of every page at a position of 1in to the right and 1in down from the top left corner of the page. By using the kernel hook directly you can put your material directly to the right place, i.e., use other coordinates in the `\put` statement above.

\AddThispageHook `\AddThispageHook{\<code>} ≡ \AddToHookNext{shipout/background}{\put(1in,-1in){\<code>}}`

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 of definition for \g__shipout_debug_bool.)

```
\shipout_debug_on: Turns debugging on and off by redefining \__shipout_debug:n.
\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 of definition for \shipout_debug_on: and others. These functions are documented on page 961.)

\ShipoutBox The box filled with the page to be shipped out (both L3 and L^AT_EX 2_E name).
\l_shipout_box

```
23 \box_new:N \l_shipout_box
24 \cs_set_eq:NN \ShipoutBox \l_shipout_box
```

(End of definition for \ShipoutBox and \l_shipout_box. These functions are documented on page 957.)

\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 of definition for \l__shipout_raw_box.)

__shipout_finalize_box: For LuaTeX invoke the `pre_shipout_filter` callback.

```

26 \sys_if_engine_luatex:TF
27 {
28   \newprotectedluacmd \_\_shipout_finalize_box:
29   \exp_args:Nx \everyjob {
30     \exp_not:V \everyjob
31     \exp_not:N \lua_now:n {
32       luatexbase.create_callback('pre_shipout_filter', 'list')
33       local~call, getbox, setbox = luatexbase.call_callback, tex.getbox, tex.setbox~
34       lua.get_functions_table() [\the \allocationnumber] = function()
35         local~head = getbox(\the \l_shipout_box)
36         local~result = call('pre_shipout_filter', head)
37         if~not (result == head) then~
38           setbox(\the \l_shipout_box, result~or~nil)
39         end~
40       end
41     }
42   }
43 } {
44   \cs_set_eq:NN \_\_shipout_finalize_box: \scan_stop:
45 }
```

(End of 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 of definition for `__shipout_execute:..`)

\shipout Overloading the `\shipout` primitive:

```
52 \cs_gset_eq:NN \shipout \_\_shipout_execute:
```

(End of definition for `\shipout`. This function is documented on page 956.)

\l__shipout_group_level_tl Helper token list to record the group level at which `__shipout_execute:` is encountered.

```
53 \tl_new:N \l__shipout_group_level_tl
```

(End of definition for `\l__shipout_group_level_tl`.)

__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_tl < \tex_currentgrouplevel:D
57   \tex_aftergroup:D \_\_shipout_execute_cont:
58 }
```

(End of definition for `__shipout_execute_test_level:..`)

__shipout_execute_cont: This does the actual shipout running several hooks as part of it. The code for them is passed as argument #2 to #4 to __shipout_execute_main_cont:Nnnn; the first argument is the box to be shipped out.

```

59 \cs_new:Npn \_\_shipout_execute_cont: {
60   \_\_shipout_execute_main_cont:Nnnn
61   \l_shipout_box
62   { \hook_use:n {shipout/before} }
63   { \hook_if_empty:nF {shipout/foreground}
64     { \_\_shipout_add_foreground_picture:n
65       { \hook_use:n {shipout/foreground} } } }
```

If the user hook for the background (shipout/background) has no code, there might still code in the kernel hook so we need to test for this too. We only test for the @kernel@before@shipout@background though. If the @kernel@after@shipout@background needs executing even if the user hook is empty then we can add another test (or the kernel could put something into the before hook).

```

66   \bool_lazy_and:nnF
67   { \hook_if_empty_p:n {shipout/background} }
68   { \tl_if_empty_p:N \@kernel@before@shipout@background }
69   { \_\_shipout_add_background_picture:n
70     { \@kernel@before@shipout@background
71       \hook_use:n {shipout/background}
72       \@kernel@after@shipout@background } }
73   }
74   }
75   { \hook_use:n {shipout/after} }
76 }
```

(End of 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^AT_EX 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.⁵⁰

```

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
```

⁵⁰In that case we don’t reset the deadcycles, that would be up to the OR processing logic to do.

```

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.⁵¹

```

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 TeX'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 %
93 %      \group_begin:
94 %      \box_set_eq_drop:NN #1 #1
95 %      \group_end:
96      \cs_set_eq:NN \protect \exp_not:N
}

```

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.⁵²

```

101     {

```

⁵¹This is not really necessary as the code could access them via `\box_ht:N`, etc., but it is perhaps convenient.

⁵²Doing that earlier would be wrong because we might end up with the last page counted but discard and then we have no place to add the final objects into the output file.

```

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:~ \@abspage@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 \@abspage@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}
119                     \@kernel@after@shipout@lastpage }

```

We record that we have handled the `shipout/lastpage` hook but only if we really did.

```

120         \bool_gset_true:N \g__shipout_lastpage_handled_bool
121     }
122 }
123 \hook_use:n {shipout}
124 \__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.

```

125     \cs_set_eq:NN \protect \exp_not:N
126     \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.

```
127          \__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.

```
128          \set@typeset@protect
129          #4
130      }
131 }
```

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).

```
132          \cs_set_eq:NN \protect \__shipout_saved_protect:
133      }
134 }
```

(End of 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.

```
135 \cs_set_protected:Npn \__shipout_execute_raw: {
136   \tl_set:Nx \l__shipout_group_level_tl
137   { \int_value:w \tex_currentgrouplevel:D }
138   \tex_afterassignment:D \__shipout_execute_test_level_raw:
139   \tex_setbox:D \l__shipout_raw_box
140 }
141 \cs_new:Npn \__shipout_execute_test_level_raw: {
142   \int_compare:nNnT
143     \l__shipout_group_level_tl < \tex_currentgrouplevel:D
144     \tex_aftergroup:D \__shipout_execute_nohooks_cont:
145 }
```

Well, not totally empty arguments, we add some debugging if we are actually doing a shipout.

```
146 \cs_new:Npn \__shipout_execute_nohooks_cont: {
147   \__shipout_execute_main_cont:Nnnn \l__shipout_raw_box
148   {} { \__shipout_debug:n{ \typeout{Doing~ raw~ shipout~ ...} }
149     \box_set_eq:NN \l__shipout_box \l__shipout_raw_box } {}
150 }
```

(End of definition for `__shipout_execute_raw:` and `__shipout_execute_test_level_raw:..`)

\RawShipout The interface name for raw shipout.

```
151 \cs_gset_eq:NN \RawShipout \__shipout_execute_raw:
```

(End of definition for `\RawShipout`. This function is documented on page 956.)

`__shipout_saved_protect:` Remember the current `\protect` state.

```
152 \cs_new_eq:NN \__shipout_saved_protect: \protect
```

(End of definition for `__shipout_saved_protect`.)

```
shipout/before  
    shipout  
shipout/after  
shipout/foreground  
shipout/background  
shipout/firstpage  
shipout/lastpage
```

Declaring all hooks for the shipout code.

```
153 \hook_new:n{shipout/before}  
154 \hook_new:n{shipout}  
155 \hook_new:n{shipout/after}  
156 \hook_new:n{shipout/foreground}  
157 \hook_new:n{shipout/background}  
158 \hook_new:n{shipout/firstpage}  
159 \hook_new:n{shipout/lastpage}
```

(End of definition for `shipout/before` and others. These functions are documented on page 957.)

\@kernel@after@shipout@lastpage
\@kernel@before@shipout@background
\@kernel@after@shipout@background

And here are the internal kernel hooks going before or after the public ones where needed.

```
160 \let@\kernel@after@shipout@lastpage\@empty  
161 \let@\kernel@before@shipout@background\@empty  
162 \let@\kernel@after@shipout@background\@empty
```

(End of 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.

```
163 \cs_new:Npn \__shipout_run_firstpage_hook: {  
164     \hook_if_empty:nTF {shipout/firstpage}
```

If not then we define the other two commands to do nothing.

```
165     {  
166         \cs_gset_eq:NN \__shipout_add_firstpage_specials: \prg_do_nothing:  
167         \cs_gset_eq:NN \__shipout_drop_firstpage_specials: \prg_do_nothing:  
168     }
```

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).

```
169     {  
170         \hbox_set:Nn \l__shipout_firstpage_box { \UseHook{shipout/firstpage} }  
171     }
```

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.

```

172  \cs_gset_eq:NN \__shipout_run_firstpage_hook: \prg_do_nothing:
173  \cs_gset:Npn \__shipout_add_firstpage_material:Nn ##1 ##2 {
174      @latex@warning{ First~ page~ is~ already~ shipped~ out,~ ignoring
175          \MessageBreak \string##1 }
176  }
177 }
```

(End of definition for __shipout_run_firstpage_hook:.)

__shipout_add_firstpage_specials:
__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.

```
178 \cs_new:Npn \__shipout_add_firstpage_specials: {
```

First we make a copy of the \l_shipout_box that we can restore it later on.

```
179 \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.

```
180 \__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.

```

181 \cs_gset_eq:NN \__shipout_add_firstpage_specials: \prg_do_nothing:
182 }
```

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.

```

183 \cs_new:Npn \__shipout_drop_firstpage_specials: {
184     \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.

```

185     \cs_gset_eq:NN \__shipout_drop_firstpage_specials: \prg_do_nothing:
186 }
```

(End of definition for __shipout_add_firstpage_specials: and
__shipout_drop_firstpage_specials:.)

\l__shipout_firstpage_box The box to hold any firstpage \specials.

```
187 \box_new:N \l__shipout_firstpage_box
```

(End of definition for \l__shipout_firstpage_box.)

\g__shipout_lastpage_handled_bool A boolean to signal if we have already handled the shipout/lastpage hook.

```
188 \bool_new:N \g__shipout_lastpage_handled_bool
```

(End of 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.

```
189 \cs_new:Npn \_shipout_add_firstpage_material:Nn #1#2 {
190     \AddToHook{shipout/firstpage}{#2}
191 }
```

(End of 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.

```
192 \cs_new:Npn \_shipout_get_box_size:N #1 {
193     \dim_set:Nn \l_shipout_box_ht_dim { \box_ht:N #1 }
194     \dim_set:Nn \l_shipout_box_dp_dim { \box_dp:N #1 }
195     \dim_set:Nn \l_shipout_box_wd_dim { \box_wd:N #1 }
196     \dim_set:Nn \l_shipout_box_ht_plus_dp_dim
197         { \l_shipout_box_ht_dim + \l_shipout_box_dp_dim }
198 }
```

(End of definition for `_shipout_get_box_size:N`.)

`\l_shipout_box_ht_dim` And here are the variables set by `_shipout_get_box_size:N`.

```
199 \dim_new:N \l_shipout_box_ht_dim
200 \dim_new:N \l_shipout_box_dp_dim
201 \dim_new:N \l_shipout_box_wd_dim
202 \dim_new:N \l_shipout_box_ht_plus_dp_dim
```

(End of definition for `\l_shipout_box_ht_dim` and others. These functions are documented on page 957.)

`\g__shipout_discard_bool` Indicate whether or not the current page box should be discarded

```
203 \bool_new:N \g__shipout_discard_bool
```

(End of 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.

```
204 \box_new:N \l__shipout_tmp_box
205 \tl_new:N \l__shipout_saved_badness_tl
```

(End of definition for `\l__shipout_tmp_box` and `\l__shipout_saved_badness_tl`.)

`_shipout_add_background_box:n` In standard L^AT_EX 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.

```
206 \cs_new:Npn \_shipout_add_background_box:n #1
207 { \_shipout_get_box_size:N \l_shipout_box
```

But we start testing for a vertical box as that should be the normal case.

```
208     \box_if_vertical:NTF \l_shipout_box
209     {
```

Save current values of `\vfuzz` and `\vbadness` then change them to allow box manipulations without warnings.

```
210      \tl_set:Nx \l__shipout_saved_badness_tl
211          { \vfuzz=\the\vfuzz\relax
212              \vbadness=\the\vbadness\relax }
213          \vfuzz=\c_max_dim
214          \vbadness=\c_max_int
```

Then we reconstruct `\l_shipout_box` ...

```
215      \vbox_set_to_ht:Nnn \l_shipout_box \l_shipout_box_ht_plus_dp_dim
216          {
```

... the material in #1 is placed into a horizontal box with zero dimensions.

```
217          \hbox_set:Nn \l__shipout_tmp_box
218              { \l__shipout_saved_badness_tl #1 }
219              \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
220              \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
221              \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).

```
222          \skip_zero:N \baselineskip
223          \skip_zero:N \lineskip
224          \skip_zero:N \lineskiplimit
225          \box_use:N \l__shipout_tmp_box
226          \vbox_unpack:N \l_shipout_box
```

The `\kern` ensures that the box has no depth which is afterwards explicitly corrected.

```
227          \kern \c_zero_dim
228      }
229      \box_set_ht:Nn \l_shipout_box \l_shipout_box_ht_dim
230      \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.

```
231          \l__shipout_saved_badness_tl
232      }
233      {
```

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.

```
234      \box_if_horizontal:NT \l_shipout_box
235          {
236              \tl_set:Nx \l__shipout_saved_badness_tl
237                  { \hfuzz=\the\hfuzz\relax
238                      \hbadness=\the\hbadness\relax }
239                      \hfuzz=\c_max_dim
240                      \hbadness=\c_max_int
241                      \hbox_set_to_wd:Nnn \l_shipout_box \l_shipout_box_wd_dim
242                          {
243                              \hbox_set:Nn \l__shipout_tmp_box
244                                  { \l__shipout_saved_badness_tl #1 }
245                                  \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
246                                  \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
247                                  \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
```

```

248          \box_move_up:nn
249              \l_shipout_box_ht_dim
250              { \box_use:N \l_shipout_tmp_box }
251          \hbox_unpack:N \l_shipout_box
252      }
253      \l__shipout_saved_badness_tl
254  }
255 }
256 }
```

(End of 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.

```

257 \cs_new:Npn \_shipout_add_foreground_box:n #1
258 {
259     \box_if_vertical:NTF \l_shipout_box
260     {
261         \tl_set:Nx \l__shipout_saved_badness_tl
262             { \vfuzz=\the\vfuzz\relax
263                 \vbadness=\the\vbadness\relax }
264         \vfuzz=\c_max_dim
265         \vbadness=\c_max_int
266         \vbox_set_to_ht:Nnn \l_shipout_box \l_shipout_box_ht_plus_dp_dim
267         {
268             \hbox_set:Nn \l__shipout_tmp_box
269                 { \l_shipout_saved_badness_tl #1 }
270             \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
271             \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
272             \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
273             \skip_zero:N \baselineskip
274             \skip_zero:N \lineskip
275             \skip_zero:N \lineskiplimit
276             \vbox_unpack:N \l_shipout_box
277             \kern -\l_shipout_box_ht_plus_dp_dim
278             \box_use:N \l__shipout_tmp_box
279             \kern \l_shipout_box_ht_plus_dp_dim
280         }
281         \l__shipout_saved_badness_tl
282         \box_set_ht:Nn \l_shipout_box \l_shipout_box_ht_dim
283         \box_set_dp:Nn \l_shipout_box \l_shipout_box_dp_dim
284     }
285     {
286         \box_if_horizontal:NT \l_shipout_box
287         {
288             \tl_set:Nx \l__shipout_saved_badness_tl
289                 { \hfuzz=\the\hfuzz\relax
290                     \hbadness=\the\hbadness\relax }
291             \hfuzz=\c_max_dim
292             \hbadness=\c_max_int
293             \hbox_set_to_wd:Nnn \l_shipout_box \l_shipout_box_wd_dim
294             {
295                 \hbox_unpack:N \l_shipout_box
296                 \kern -\box_wd:N \l_shipout_box

```

```

297          \hbox_set:Nn \l__shipout_tmp_box
298              { \l__shipout_saved_badness_tl #1 }
299          \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
300          \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
301          \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
302          \box_move_up:nn { \box_ht:N \l_shipout_box }
303              { \box_use:N \l__shipout_tmp_box }
304          \kern \box_wd:N \l_shipout_box
305      }%
306      \l__shipout_saved_badness_tl
307  }
308 }
309 }
```

(End of 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.

```

310 \cs_new:Npn \__shipout_init_page_origins: {
311     \tl_const:Nx \c__shipout_horigin_tl
312     {
313         \cs_if_exist_use:NTF \pdfvariable { horigin }
314             { \cs_if_exist_use:NF \pdfhorigin { 1in } }
315     }
316     \tl_const:Nx \c__shipout_vorigin_tl
317     {
318         \cs_if_exist_use:NTF \pdfvariable { vorigin }
319             { \cs_if_exist_use:NF \pdfvorigin { 1in } }
320     }
321 }
```

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.

```

321     \cs_gset_eq:NN \__shipout_init_page_origins: \prg_do_nothing:
322 }
```

(End of 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.

```

323 \cs_new:Npn \__shipout_picture_overlay:n #1 {
```

The very first time this is executed we have to initialize (and freeze) the origins.

```

324     \__shipout_init_page_origins:
325     \kern -\c__shipout_horigin_tl \scan_stop:
326     \vbox_to_zero:n {
327         \kern -\c__shipout_vorigin_tl \scan_stop:
328         \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).

```

329      \hbox_set_to_wd:Nnn \l__shipout_tmp_box \c_zero_dim
330          { \ignorespaces #1 \hss }
331      \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
332      \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
333      \box_use:N \l__shipout_tmp_box
334      \tex_vss:D
335  }
336 }
```

(End of 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.

```

337 \cs_new:Npn \__shipout_add_background_picture:n #1 {
338     \__shipout_add_background_box:n { \__shipout_picture_overlay:n {#1} }
339 }
```

(End of 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.

```

340 \cs_new:Npn \__shipout_add_foreground_picture:n #1 {
341     \__shipout_add_foreground_box:n { \__shipout_picture_overlay:n {#1} }
342 }
```

(End of 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^AT_EX looks ahead and is not using the position for on the next page).

```

343 \cs_new_protected:Npn \shipout_discard: {
344     \bool_gset_true:N \g__shipout_discard_bool
345 }
```

(End of definition for `\shipout_discard:`. This function is documented on page 960.)

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.

```
346 \int_new:N \g_shipout_READONLY_int
```

For L^AT_EX 2_ε it is available as a command (i.e., a T_EX counter only).

```
347 \cs_new_eq:NN \ ReadonlyShipoutCounter \g_shipout_READONLY_int
```

(End of definition for `\g_shipout_READONLY_int` and `\ ReadonlyShipoutCounter`. These functions are documented on page 961.)

`\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.

348 `\int_new:N \g_shipout_totalpages_int`

For L^AT_EX 2 _{ε} this is offered as a L^AT_EX counter so can be easily typeset inside the output routine to display things like “`\thepage/\thetotalpages`”, etc.

349 `\cs_new_eq:NN \c@totalpages \g_shipout_totalpages_int`
 350 `\cs_new:Npn \thetotalpages { \arabic{totalpages} }`

(End of definition for `\g_shipout_totalpages_int` and `\c@totalpages`. These functions are documented on page 961.)

`\@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_EX. 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.

351 `\xdef\@abspage@last{\number\maxdimen}`

(End of 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`

352 `\g@addto@macro \@kernel@after@enddocument {`
 353 `\int_compare:nNnT \@abspage@last = \maxdimen`
 354 `{`

We use L^AT_EX 2 _{ε} coding as `\@abspage@last` is not an L₃ name.

355 `\xdef\@abspage@last{ \int_eval:n { \g_shipout_READONLY_int + 1 } }`
 356 `}`
 357 `}`

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.

`\@kernel@after@enddocument@afterlastpage`

358 `\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.

359 `\int_compare:nNnF \g_shipout_READONLY_int = 0`
 360 `{`

This ends up in the `.aux` so we use L^AT_EX 2 _{ε} names here.

Todo: This needs an interface for \nofiles in expl3, doesn't at the moment!

```

361      \if@filesw
362          \iow_now:Nx \@auxout {
363              \gdef\string\@abspage@last {\int_use:N \g_shipout_READONLY_int}
364      \fi

```

But we may have guessed wrongly earlier and have run it too early or 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. In either case we should put out an appropriate “rerun” warning.

```

365      \bool_if:NTF \g__shipout_lastpage_handled_bool
366          {

```

If the hook was already executed, we have to test if that total shipouts match the shipouts from last run (because that corresponds to the page it was executed). If not we output a warning.

```

367          \int_compare:nNnF \@abspage@last = \g_shipout_READONLY_int
368          {
369              \@latex@warning@no@line{Hook~ 'shipout/lastpage'~ executed-
370                  on~ wrong~ page~ (\@abspage@last\space not-
371                  \int_use:N\g_shipout_READONLY_int). \MessageBreak
372                  Rerun~ to~ correct~ this}%
373          }
374      }
375      {

```

If the hook was not run, we need to add an extra page and place it there. 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.

```

376      \bool_lazy_and:nnF
377          { \hook_if_empty_p:n {shipout/lastpage} }
378          { \tl_if_empty_p:N \@kernel@after@shipout@lastpage }
379          {
380              \tex_shipout:D\vbox to\textheight
381              {
382                  \hbox:n { \UseHook{shipout/lastpage}
383                      \@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.

```

384          \__shipout_excuse_extra_page:
385          \null
386      }

```

At this point we also signal to L^AT_EX'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`.

```

387          \cs_gset_eq:NN \@extra@page@added \relax
388          }
389      }
390  }
391 }

```

(End of definition for `\enddocument`, `\@kernel@after\enddocument`, and
`\@kernel@after\enddocument@afterlastpage`.)

```

\_shipout_excuse_extra_page: Say mea culpa ...
392 \cs_new:Npn \_shipout_excuse_extra_page: {
393   \vfil
394   \begin{center}
395     \bfseries Temporary~ page!
396   \end{center}
397   \LaTeX{} was~ unable~ to~ guess~ the~ total~ number~ of~ pages~
398   correctly.~ ~ As~ there~ was~ some~ unprocessed~ data~ that~
399   should~ have~ been~ added~ to~ the~ final~ page~ this~ extra~
400   page~ has~ been~ added~ to~ receive~ it.
401   \par
402   If~ you~ rerun~ the~ document~ (without~ altering~ it)~ this~
403   surplus~ page~ will~ go~ away,~ because~ \LaTeX{} now~ knows~
404   how~ many~ pages~ to~ expect~ for~ this~ document.
405   \vfil
406 }
```

(End of definition for `_shipout_excuse_extra_page`.)

`\PreviousTotalPages` In the preamble before the aux file was read `\PreviousTotalPages` is always zero.
`\@kernel@before\begindocument`

In the aux file there should be an update for `\@abspage@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.

```

408 \g@addto@macro\@kernel@before\begindocument
409   {\ifnum\@abspage@last<\maxdimen
410     \xdef\PreviousTotalPages{\@abspage@last}\fi}
```

(End of definition for `\PreviousTotalPages` and `\@kernel@before\begindocument`. These functions are documented on page 961.)

4 Legacy L^AT_EX 2_& interfaces

`\DiscardShipoutBox` Request that the next shipout box is to be discarded.

```

411 \cs_new_eq:NN \DiscardShipoutBox \shipout_discard:
```

(End of definition for `\DiscardShipoutBox`. This function is documented on page 960.)

`\AtBeginDvi` If we roll forward from an earlier kernel `\AtBeginDvi` is defined so we better not use `\cs_new_protected:Npn` here.

```

412 \cs_set_protected:Npn \AtBeginDvi
413   {\_shipout_add_firstpage_material:Nn \AtBeginDvi}
```

(End of definition for `\AtBeginDvi`. This function is documented on page 959.)

`\DebugShipoutsOn`

`\DebugShipoutsOff`

```

414 \cs_new_eq:NN \DebugShipoutsOn \shipout_debug_on:
415 \cs_new_eq:NN \DebugShipoutsOff \shipout_debug_off:
```

(End of definition for `\DebugShipoutsOn` and `\DebugShipoutsOff`. These functions are documented on page 961.)

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.

416 ⟨@@=⟩

Some internals needed elsewhere.

```
417 \cs_set_eq:NN \cexpl@@@shipout@add@firstpage@material@@Nn
418           \__shipout_add_firstpage_material:Nn
419 \cs_set_eq:NN \cexpl@@@shipout@add@background@box@@n
420           \__shipout_add_background_box:n
421 \cs_set_eq:NN \cexpl@@@shipout@add@foreground@box@@n
422           \__shipout_add_foreground_box:n
423 \cs_set_eq:NN \cexpl@@@shipout@add@background@picture@@n
424           \__shipout_add_background_picture:n
425 \cs_set_eq:NN \cexpl@@@shipout@add@foreground@picture@@n
426           \__shipout_add_foreground_picture:n
```

(End of definition for \cexpl@@@shipout@add@firstpage@material@@Nn and others.)

```
427 \ExplSyntaxOff
428 ⟨/ekernel | latexrelease⟩
429 ⟨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.

```
430 ⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
431 ⟨latexrelease⟩           {\shipout}{Hook management (shipout)}%
432 ⟨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!

```
433 ⟨latexrelease⟩\ifcsname tex_shipout:D\endcsname
434 ⟨latexrelease⟩\expandafter\let\expandafter\shipout
435 ⟨latexrelease⟩           \csname tex_shipout:D\endcsname
436 ⟨latexrelease⟩\fi
437 ⟨latexrelease⟩
438 ⟨latexrelease⟩\let \RawShipout\@undefined
439 ⟨latexrelease⟩\let \ShipoutBox\@undefined
440 ⟨latexrelease⟩\let \ ReadonlyShipoutCounter \@undefined
441 ⟨latexrelease⟩\let \c@totalpages \@undefined
442 ⟨latexrelease⟩\let \thetotalpages \@undefined
443 ⟨latexrelease⟩
444 ⟨latexrelease⟩\let \DiscardShipoutBox \@undefined
445 ⟨latexrelease⟩\let \DebugShipoutsOn \@undefined
446 ⟨latexrelease⟩\let \DebugShipoutsOff \@undefined
447 ⟨latexrelease⟩
448 ⟨latexrelease⟩\DeclareRobustCommand \AtBeginDvi [1]{%
449 ⟨latexrelease⟩ \global \setbox \@begindvibox
450 ⟨latexrelease⟩ \vbox{\unvbox \@begindvibox #1}%
451 ⟨latexrelease⟩}
```

```

452 〈\latexrelease〉
453 〈\latexrelease〉\let \AtBeginShipout \@undefined
454 〈\latexrelease〉\let \AtBeginShipoutNext \@undefined
455 〈\latexrelease〉
456 〈\latexrelease〉\let \AtBeginShipoutFirst \@undefined
457 〈\latexrelease〉
458 〈\latexrelease〉\let \ShipoutBoxHeight \@undefined
459 〈\latexrelease〉\let \ShipoutBoxDepth \@undefined
460 〈\latexrelease〉\let \ShipoutBoxWidth \@undefined
461 〈\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).

```

462 〈\latexrelease〉
463 〈\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).

```

464 %\reenable@package@load{atenddvi}
465 〈\latexrelease〉
466 〈\latexrelease〉\EndIncludeInRelease
467 〈*2ekernel〉

```

6 Package emulation for compatibility

6.1 Package `atenddvi` emulation

`\AtEndDvi` This package has only one public command, so simulating it is easy and actually sensible to provide as part of the kernel.

```

468 〈/2ekernel〉
469 〈*2ekernel | latexrelease〉
470 〈\latexrelease〉\IncludeInRelease{2020/10/01}%
471 〈\latexrelease〉\AtEndDvi{atenddvi emulation}%
472 \ExplSyntaxOn
473 \cs_new_protected:Npn \AtEndDvi #1 {\AddToHook{shipout/lastpage}{#1}}
474 \ExplSyntaxOff

```

As the package is integrate we prevent loading (no need to roll that back):

```

475 \disable@package@load{atenddvi}
476 \PackageWarning{atenddvi}
477 {Functionality of this package is already\MessageBreak
478 provided by LaTeX.\MessageBreak\MessageBreak
479 It is there no longer necessary to load it.\MessageBreak
480 and you can safely remove it.\MessageBreak
481 Found on}%
482 〈/2ekernel | latexrelease〉

```

```

483 〈\latexrelease〉\EndIncludeInRelease
484 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
485 〈\latexrelease〉                                {＼AtEndDvi}{atenddvi emulation}%
486 〈\latexrelease〉\let \AtEndDvi \undefined
487 〈\latexrelease〉\EndIncludeInRelease
488 {*2ekernel}

(End of definition for \AtEndDvi. This function is documented on page 959.)

489 ⟨/2ekernel⟩

```

6.2 Package **atbegshi** emulation

```

490 ⟨*atbegshi-ltx⟩
491 \ProvidesPackage{atbegshi-ltx}
492 [2021/01/10 v1.0c
493   Emulation of the original atbegshi^Jpackage with kernel methods]

```

\AtBeginShipoutBox

```
494 \let \AtBeginShipoutBox \ShipoutBox
```

(*End of definition for \AtBeginShipoutBox. This function is documented on page 962.*)

\AtBeginShipoutInit

Compatibility only, we aren't delaying ...

```
495 \let \AtBeginShipoutInit \empty
```

(*End of definition for \AtBeginShipoutInit. This function is documented on page 963.*)

\AtBeginShipout

Filling hooks

```

496 \protected\long\def\AtBeginShipout      #1{\AddToHook{shipout/before}{#1}}
497 \protected\long\def\AtBeginShipoutNext #1{\AddToHookNext{shipout/before}{#1}}

```

(*End of definition for \AtBeginShipout and \AtBeginShipoutNext. These functions are documented on page 963.*)

\AtBeginShipoutFirst

Slightly more complex as we need to know the name of the command under which the shipout/firstpage hook is filled.

```

498 \protected \def \AtBeginShipoutFirst
499   {\Expl@@@shipout@add@firstpage@material@Nn \AtBeginShipoutFirst}

```

(*End of definition for \AtBeginShipoutFirst. This function is documented on page 963.*)

\AtBeginShipoutDiscard

Just a different name.

```
500 \let \AtBeginShipoutDiscard \DiscardShipoutBox
```

(*End of definition for \AtBeginShipoutDiscard. This function is documented on page 963.*)

\AtBeginShipoutAddToBox

We don't expose them.

```

501 \let \AtBeginShipoutAddToBox
502   \Expl@@@shipout@add@background@box@Nn
503 \let \AtBeginShipoutAddToBoxForeground
504   \Expl@@@shipout@add@foreground@box@Nn
505 \let \AtBeginShipoutUpperLeft
506   \Expl@@@shipout@add@background@picture@Nn
507 \let \AtBeginShipoutUpperLeftForeground
508   \Expl@@@shipout@add@foreground@picture@Nn

```

(End of definition for \AtBeginShipoutAddToBox and others. These functions are documented on page 962.)

\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.

```
509 \ExplSyntaxOn
510 \cs_new_eq:NN \AtBeginShipoutOriginalShipout \tex_shipout:D
```

(End of definition for \AtBeginShipoutOriginalShipout. This function is documented on page 962.)

\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.

```
511 \cs_new:Npn \ShipoutBoxHeight { \dim_use:N \l_shipout_box_ht_dim }
512 \cs_new:Npn \ShipoutBoxDepth { \dim_use:N \l_shipout_box_dp_dim }
513 \cs_new:Npn \ShipoutBoxWidth { \dim_use:N \l_shipout_box_wd_dim }
514 \ExplSyntaxOff
```

(End of definition for \ShipoutBoxHeight, \ShipoutBoxWidth, and \ShipoutBoxDepth.)

```
515 </atbegshi-ltx>
```

If the package is requested we substitute the one above:

```
516 <*2ekernel>
517 \declare@file@substitution{atbegshi.sty}{atbegshi-ltx.sty}
518 </2ekernel>
```

6.3 Package **everyshi** emulation

This is now directly handled in that package so emulation is not necessary any more.

Rather important :-)

```
519 <@@=
```

File Y

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^AT_EX 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> <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 \@dbltoplist 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 \@botlist, 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 \@botlist, 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 \@dbltoplist 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^AT_EX 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 of definition for `\AtBeginDvi` and `\@begindvibox`. These functions are documented on page 959.)

\@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 of definition for `\@maxdepth`.)

\paperheight New `\paper...` registers.

```

93 \newdimen\paperheight
94 \newdimen\paperwidth

```

(End of definition for `\paperheight` and `\paperwidth`.)

\stockheight New `\stock...` registers.

```

95 \newdimen\stockheight
96 \newdimen\stockwidth

```

(End of definition for `\stockheight` and `\stockwidth`.)

\if@insert Local switches first:

```

97 \newif \if@insert

```

These should definitely be global:

```

98 \newif \if@fcollmade
99 \newif \if@specialpage \@specialpagefalse

```

These should be global but are not always set globally in other files.

```

100 \newif \if@firstcolumn \@firstcolumntrue
101 \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?

```
102 \newif \if@twoside      \@twosidefalse
103 \newif \if@reversemargin \@reversemarginfalse
104 \newif \if@mparswitch   \@mparswitchfalse
```

This counter has been imported from ‘multicol’.

```
105 \newcount \col@number
106 \col@number \one
```

(End of definition for `\if@insert` and others.)

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

INTERNAL REGISTERS

```
107 \newcount\@topnum
108 \newdimen\@toproom
109 \newcount\@dbltopnum
110 \newdimen\@dbltoproom
111 \newcount\@botnum
112 \newdimen\@botroom
113 \newcount\@colnum
114 \newdimen\@textmin
115 \newdimen\@fpmin
116 \newdimen\@colht
117 \newdimen\@colroom
118 \newdimen\@pageht
119 \newdimen\@pagedp
120 \newdimen\@mparbottom \@mparbottom\z@
121 \newcount\@currtype
122 \newbox\@outputbox
123 \newbox\@leftcolumn
124 \newbox\@holdpg
```

```
125 \def\@thehead{\@oddhead} % initialization
126 \def\@thefoot{\@oddfoot}
```

End of historical L^AT_EX 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.

```
127 \def\clearpage{%
128   \ifvmode
129     \ifnum \@dbltopnum =\m@ne
130       \ifdim \pagetotal <\topskip
131         \hbox{}%
132       \fi
133     \fi
134   \fi
135   \newpage
136   \write\m@ne{}}%
```

```
137     \vbox{}}%
138     \penalty -\@Mi
139 }

(End of definition for \clearpage.)
```



```
140 \def\cleardoublepage{\clearpage\if@twoside \ifodd\c@page\else
141     \hbox{}\newpage\if@twocolumn\hbox{}\newpage\fi\fi\fi}
142 </2ekernel>

(End of definition for \cleardoublepage.)
```

```
\onecolumn

143  <*ekernel | fltrace>
144  \def\onecolumn{%
145    \clearpage
146    \global\columnwidth\textwidth
147    \global\hsize\columnwidth
148    \global\linewidth\columnwidth
149    \global\@twocolumnfalse
150    \col@number \@ne
151    \@floatplacement}
```

(End of 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`.

```
152 </2ekernel | fltrace>
153 <|latexrelease>\IncludeInRelease{2017/04/15}%
154 <|latexrelease>           {\newpage}{Check depth of page}%
155 <*2ekernel | latexrelease | fltrace>
156 \def \newpage {%
157   \if@noskipsec
158     \ifx \c@nodocument\relax
159       \leavevmode
160     \global \c@noskipsecfalse
161   \fi
162 \fi
163 \if@inlabel
164   \leavevmode
165   \global \c@inlabelfalse
166 \fi
167 \if@nobreak \c@nobreakfalse \everypar{}\fi
168 \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.

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`.

```

169  \ifdim\prevdepth>\z@%
170    \vskip -%
171    \ifdim\prevdepth>\maxdepth
172      \maxdepth
173    \else
174      \prevdepth
175    \fi
176  \fi
177  \vfil
178  \penalty -\@M}
179 </2ekernel | latexrelease | fltrace>
180 <latexrelease>\EndIncludeInRelease
181 <latexrelease>\IncludeInRelease{0000/00/00}%
182 <latexrelease>          {\newpage}{Check depth of page}%
183 <latexrelease>\def \newpage {%
184 <latexrelease>  \if@noskipsec
185 <latexrelease>    \ifx \@nodocument\relax
186 <latexrelease>      \leavevmode
187 <latexrelease>      \global \c@noskipsecfalse
188 <latexrelease>    \fi
189 <latexrelease>  \fi
190 <latexrelease>  \if@inlabel
191 <latexrelease>    \leavevmode
192 <latexrelease>    \global \c@inlabelfalse
193 <latexrelease>  \fi
194 <latexrelease>  \if@nobreak \c@nobreakfalse \everypar{}\fi
195 <latexrelease>  \par
196 <latexrelease>  \vfil
197 <latexrelease>  \penalty -\@M}
198 <latexrelease>\EndIncludeInRelease
199 <*2ekernel | fltrace>
```

(End of definition for `\newpage`.)

`\@emptycol` It may be better to use an invisible rule rather than an empty box here.

```
200 \def \@emptycol {\vbox{} \penalty -\@M}
```

(End of definition for `\@emptycol`.)

`\twocolumn` There are several bug fixes to the two-column stuff here.

```

201 \def \twocolumn {%
202   \clearpage
203   \global \columnwidth \textwidth
204   \global \advance \columnwidth -\columnsep
205   \global \divide \columnwidth \tw@
206   \global \hsize \columnwidth
207   \global \linewidth \columnwidth
208   \global \c@twocolumntrue
209   \global \c@firstcolumntrue
210   \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.

```
211  \@ifnextchar [\@topnewpage\@floatplacement
212 }
```

Note that here, getting a box from the freelist can assume success since this comes just after a `\clearpage`.

```
213 \long\def \@topnewpage [#1]{%
214   \@nodocument
215   \@next\currbox\@freelist{}{}%
216   \global \setbox\currbox
217     \color@vbox
218       \normalcolor
219       \vbox{%
220         \hsize\textwidth
221         \parboxrestore
222         \col@number \one
223         #1%
224         \vskip -\dbltextfloatsep
225       }%
226     \color@endbox
```

Added size test and warning message; perhaps we should use an error message.

```
227   \ifdim \ht\currbox>\textheight
228     \ht\currbox \textheight
229   \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.

```
230   \global \count\currbox \tw@
231   \@tempdima -\ht\currbox
232   \advance \atempdima -\dbltextfloatsep
233   \global \advance \colht \atempdima
234   \ifx \dbltoplist \empty
235   \else
236     \@latex@error{Float(s) lost}\ehb
237     \let \dbltoplist \empty
238   \fi
239   \cons \dbltoplist \currbox
```

This setting of `\@dbltopnum` is used only to change the typesetting in `\@combinedblfloats`.

```
240   \global \dbltopnum \m@ne
241   <*trace>
242     \f@trace{dbltopnum set to -1 (= \the \dbltopnum) (topnewpage)}%
243   </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.

```

244 \ifdim \@colht<2.5\baselineskip
245   \@latex@warning@no@line {Optional argument of \noexpand\twocolumn
246   too tall on page \thepage}%
247   \@emptycol
248   \if@firstcolumn
249   \else
250     \@emptycol
251   \fi
252   \else
253     \global \vsize \@colht
254     \global \colroom \@colht
255     \@floatplacement
256   \fi
257 }
```

(End of 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.

```

258 \output {%
259   \let \par \@@par
260   \ifnum \outputpenalty<-`@M
261     \@specialoutput
262   \else
263     \@makecol
264     \@opcol
```

Moved to `\@opcol`: `\@floatplacement`.

```
265   \@startcolumn
```

This loop could be replaced by an `\expandafter` tail recursion in `\@startcolumn`.

```

266   \@whilesw \if@fcolmade \fi
267   {%
268   (*trace)
269     \f1@trace{PAGE: float \if@twocolumn column \else page \fi
270               completed}%
271   (/trace)
272   \@opcol\@startcolumn}%
273   \fi
274   \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.

```

275      \ifdim \@colroom<1.5\baselineskip
276          \ifdim \@colroom<\textheight
277              \@latex@warning@no@line {Text page \thepage\space
278                  contains only floats}%
279          \@emptycol
280      %
281          \if@twocolumn
282          %
283          \else
284          %
285          \fi
286      %
287          \global \vsize \@colroom
288      %
289      \else
290          \global \vsize \@colroom
291      %
292      \else
293          \global \vsize \maxdimen
294      %
295  }
```

Historical L^AT_EX 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^AT_EX 2.09 comments.

```

296  \gdef\@specialoutput{%
297      \ifnum \outputpenalty>-\@Mii
298          \@doclearpage
```

```

299     \else
300         \ifnum \outputpenalty<-\@Mii
301             \ifnum \outputpenalty<-\@MM \deadcycles \z@ \fi
302             \global \setbox\@holdpg \vbox {\unvbox\@cclv}%
303         \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.

```

304     \global \setbox\@holdpg \vbox{%
305         \unvbox\@holdpg
306         \unvbox\@cclv

```

We must now remove the box added by the float mechanism and the `\topskip` glue therefore added above it by TeX.

```

307     \setbox\@tempboxa \lastbox
308     \unskip
309     }%

```

These two are needed as separate dimensions only by `\@addmarginpar`; for other purposes we put the whole size into `\@pageht` (see below).

```

310     \@pagedp \dp\@holdpg
311     \@pageht \ht\@holdpg
312     \unvbox \@holdpg
313     \next\currbox\@currlist{%
314         \ifnum \count\currbox>\z@

```

Putting the whole size into `\@pageht` (see above).

```

315     \advance \@pageht \@pagedp
316     \ifvoid\footins \else
317         \advance \@pageht \ht\footins
318         \advance \@pageht \skip\footins
319         \advance \@pageht \dp\footins
320     \fi
321     \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?

```

322         \ifdim \wd\@kludgeins=\z@
323             \advance \@pageht \ht\@kludgeins
324         (*trace)
325             \f@trace {Extra size added: \the \ht\@kludgeins}%
326     (/trace)
327         \fi
328         \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.

```

329          \@reinserts
330          \@addtocurcol
331      \else
332          \@reinserts
333          \@addmarginpar
334      \fi
335  }\@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.

```

336 \ifnum \outputpenalty<\z@
337   \if@nobreak
338     \nobreak
339   \else
340     \addpenalty \interlinepenalty
341   \fi
342   \fi
343   \fi
344 \fi
345 }
346 </2ekernel | fltrace>

```

(End of 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).

```

347 <latexrelease>\IncludeInRelease{2015/01/01}%
348 <latexrelease>      {\@testwrongwidth}{float order in 2-column}%
349 <*2ekernel | latexrelease | fltrace>

350 \def\@testwrongwidth #1{%
351   \ifdim\dp#1=\f@depth
352   {*trace}
353     \f1@trace{\string#1
354       \ifdim\f@depth=\z@ single \else double \fi
355       column float -- ok}%
356   /trace}
357   \else
358     \global\@testtrue
359   {*trace}
360     \f1@trace{\string#1
361       \ifdim\f@depth=\z@ double \else single \fi
362       column float -- wrong}%
363   /trace}
364   \fi}%

```

Normally looking for single column floats, which have zero depth.
`\let\f@depth\z@`

```

366  {/2ekernel | latexrelease | fltrace}
367  \end{IncludeInRelease}
368  \IncludeInRelease{0000/00/00}%
369  \begin{IncludeInRelease}
370    {\@testwrongwidth}{float order in 2-column}%
371  \let\@testwrongwidth\undefined
372  \let\f@depth\undefined
373  \end{IncludeInRelease}

```

(End of 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.

```

373  \IncludeInRelease{2015/01/01}{\@doclearpage}%
374  \begin{IncludeInRelease}
375    {*2ekernel | latexrelease}
376    \def \@doclearpage {%
377      \ifvoid\footins
378        \ifvbox\@kludgeins
379          {\setbox \tempboxa \box \kludgeins}%
380      \else
381        \f@trace {kludgeins box made void}%
382      \fi
383      \setbox\tempboxa\vsplit\cclv to\z@\unvbox\tempboxa
384      \setbox\tempboxa\box\cclv
385      \xdef\@deferlist{\@toplist\@botlist\@deferlist}%
386      \global \let \@toplist \empty
387      \global \let \@botlist \empty
388      \global \colroom \colht
389      \ifx \currlist\empty
390        \else
391          \@latex@error{Float(s) lost}\ehb
392          \global \let \currlist \empty
393        \fi
394      \makefcolumn\@deferlist
395      \whilesw\if@fcolmade \fi{\opcol\makefcolumn\@deferlist}%
396      \if@twocolumn
397        \if@firstcolumn
398          \xdef\@deferlist{\dbltoplist\@deferlist}%
399          \global \let \dbltoplist \empty
400          \global \colht \textheight
401          \begingroup
402            \dblfloplacement

```

```

404     \@makefcolumn\@deferlist
405     \@whilesw\if@fcolmade \fi{\@outputpage
406                               \@makefcolumn\@deferlist}%
407
408     \endgroup
409     \else
410         \vbox{}\clearpage
411     \fi
411 
```

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.

```

412     \ifx\@deferlist\@empty \else\clearpage \fi
413     \else
414         \setbox\@cclv\vbox{\box\@cclv\vfil}%
415         \@makecol\@opcol
416         \clearpage
417     \fi
418 }%
419 </2ekernel | latexrelease>
420 <latexrelease>\EndIncludeInRelease
421 <latexrelease>\IncludeInRelease{0000/00/00}{\@doclearpage}%
422 <latexrelease>                                {float order in 2-column}%
423 <latexrelease>\def \@doclearpage {%
424 <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.

```

425 <latexrelease>      \ifvbox\@kludgeins
426 <latexrelease>          {\setbox \@tempboxa \box \@kludgeins}%
427 <*trace>
428 <latexrelease>          \f1@trace {\kludgeins box made void}%
429 </trace>
430 <latexrelease>          \fi
431 <latexrelease>          \setbox\@tempboxa\vsplit\@cclv to\z@\unvbox\@tempboxa
432 <latexrelease>          \setbox\@tempboxa\box\@cclv
433 <latexrelease>          \xdef\@deferlist{\@toplist\@botlist\@deferlist}%

434 <latexrelease>          \global \let \@toplist \@empty
435 <latexrelease>          \global \let \@botlist \@empty
436 <latexrelease>          \global \@colroom \@colht
437 <latexrelease>          \ifx \@currlist\@empty
438 <latexrelease>          \else
439 <latexrelease>              \@latexerr{Float(s) lost}\@ehb

440 <latexrelease>          \global \let \@currlist \@empty
441 <latexrelease>          \fi
442 <latexrelease>          \@makefcolumn\@deferlist
443 <latexrelease>          \@whilesw\if@fcolmade \fi
444                               {\@opcol\@makefcolumn\@deferlist}%
445 <latexrelease>          \if@twocolumn
446 <latexrelease>              \if@firstcolumn
447 <latexrelease>                  \xdef\@dbldeferlist{\@dbltoplist\@dbldeferlist}%
 
```

```

448 <|latexrelease>          \global \let \@dbltoplist \empty
449 <|latexrelease>          \global \@colht \textheight
450 <|latexrelease>          \begingroup
451 <|latexrelease>          \@dblfloatingplacement
452 <|latexrelease>          \@makefcolumn\@dbldeflist
453 <|latexrelease>          \@whilsw\if@fcolmade \fi
454 <|latexrelease>          {\@outputpage\@makefcolumn\@dbldeflist}%
455 <|latexrelease>          \endgroup
456 <|latexrelease>          \else
457 <|latexrelease>          \vbox{}\clearpage
458 <|latexrelease>          \fi
459 <|latexrelease>          \fi
460 <|latexrelease>          \else
461 <|latexrelease>          \setbox\@cclv\vbox{\box\@cclv\vfil}%
462 <|latexrelease>          \@makecol\@opcol
463 <|latexrelease>          \clearpage
464 <|latexrelease>          \fi
465 <|latexrelease>  }%
466 <|latexrelease>\EndIncludeInRelease

```

(End of definition for \@doclearpage.)

\@opcol Several changes in detail here.

```

467 <|*2ekernel | fltrace>
468 \def \@opcol {%
469   \if@twocolumn

```

The funny-looking internal commands are interfacing with the new marks mechanism. We make sure (elsewhere) that those are always defined, even when we roll back, so here we add them unconditionally. This still need turning into a hook or config point eventually:

```

470   \expl@@mark@update@dblcol@structures@@
471   \outputdblcol
472 \else
473   \expl@@mark@update@singlcol@structures@@
474   \outputpage
475 <|*trace>
476   \fl@trace{PAGE: one column (float? see above) page completed}%
477 </trace>

```

Not needed since it comes after \@outputpage:

```

478 %   \global\@colht\textheight
479 \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.

```

480   \global \z@ \global \textfloatsheight \z@
481   \floatplacement
482 }
483 <|/2ekernel | fltrace>

```

(End of 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.

```
484 {*2ekernel}
485 \gdef \@makecol {%
486   \ifvoid\footins
487     \setbox\@outputbox \box\@cclv
488   \else
489     \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.

```
490   \boxmaxdepth \@maxdepth
491   %
492   \unvbox \@cclv
493   %
494   \vskip-\@tempdima
495   \vskip \skip\footins
496   \color@begingroup
497   \normalcolor
498   \footnoterule
499   \unvbox \footins
500   \color@endgroup
501 }%
502 \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.

```
502 \let\@elt\relax
503 \xdef\@freelist{\@freelist\@midlist}%
504 \global \let \@midlist \empty
505 \combinefloats
```

The variations start here in case \enlargethispage has been used.

```
506 \ifvbox\@kludgeins
507   \makespecialcolbox
508 \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 -\dimen@ 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`.

```
509      \setbox\@outputbox \vbox to\@colht {%
510  %      \boxmaxdepth \maxdepth                      %??
511      \@texttop
512      \dimen0 \dp\@outputbox
513      \unvbox\@outputbox
514      \vskip -\dimen0
515      \@textbottom
516  }%
517  \fi
518  \global \maxdepth \@maxdepth
519 }
```

(End of 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.

```
520 \gdef \@reinserts{%
521   \ifvoid\footins\else\insert\footins{\unvbox\footins}\fi
522   \ifvbox\@kludgeins\insert\@kludgeins
523     {\unvbox\@kludgeins}\fi
524 }
525 
```

(End of definition for `\@reinserts`.)

`\@makespecialcolbox` This implements certain variations in page-makeup.

```
526 (*2ekernel | fltrace)
527 \gdef \@makespecialcolbox {%
528 <*trace>
529   \fl@trace{Kludgeins ht \the\ht\@kludgeins\space
530             dp \the\dp\@kludgeins\space
531             wd \the\wd\@kludgeins}%
532 }
```

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’.

```
533 \setbox\@outputbox \vbox {%
534   \@texttop
535   \dimen0 \dp\@outputbox
536   \unvbox\@outputbox
537   \vskip-\dimen0
538 }%
539 \tempdima \@colht
540 \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 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.

There 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.

```

541      \advance \@tempdima -\ht\@outputbox
542      \advance \@tempdima \pageshrink
543  {*trace}
544      \f@trace {Natural ht of col: \the \ht\@outputbox}%
545      \f@trace {\string \@colht: \the \@colht}%
546      \f@trace {Pageshrink added: \the \pageshrink}%
547      \f@trace {Hence, space added: \the \@tempdima}%
548  {/trace}
549      \setbox\@outputbox \vbox to \@colht {%
550  %
551      \boxmaxdepth \maxdepth
552      \unvbox\@outputbox
553      \vskip \@tempdima
554      \textbottom
555  }%

```

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.

```

555  \else
556      \advance \@tempdima -\ht\@kludgeins
557  {*trace}
558      \f@trace {Natural ht of col: \the \ht\@outputbox}%
559      \f@trace {\string \@colht: \the \@colht}%
560      \f@trace {Extra size added: -\the \ht \@kludgeins}%
561      \f@trace {Hence, height of inner box: \the \@tempdima}%
562      \f@trace {Max? pageshrink available: \the \pageshrink}%
563  {/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.

```

564      \setbox \@outputbox \vbox to \@colht {%
565          \vbox to \@tempdima {%

```

```

566          \unvbox\@outputbox
567          \@textbottom}%
568          \vss}%
569      \fi

```

Finally we need to explicitly make the insert box void.

```

570      {\setbox \@tempboxa \box \@kludgeins}%
571      {*trace}
572          \f1@trace {kludgeins box made void}%
573      {/trace}
574  }
575  {/2ekernel | ftrace}

```

(*End of definition for \@makespecialcolbox.*)

\@texttop These do nothing as a default.
\@textbottom
576 {*2ekernel}
577 \let \@texttop \relax
578 \let \@textbottom \relax

(*End of 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.

```

579 \def\@activechar@info #1{%
580     \clatex@info@no@line {Active #1 character found while
581                             output routine is active
582                             \MessageBreak
583                             This may be a bug in a package file
584                             you are using}%
585 }

```

Do not put any spaces in this next bit!

```

586 \begingroup
587 \obeylines\obeyspaces%
588 \catcode`\'\active%
589 \gdef\@resetactivechars{%
590 \def^~M{\@activechar@info{EOL}\space}%
591 \def {\@activechar@info{space}\space}%
592 \let'\active@math@prime}%
593 \endgroup

```

(*End of 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.

```
594  {/2ekernel}
595  <latexrelease>\IncludeInRelease{2017/04/15}%
596  <latexrelease> {\@outputpage}{Reset language for hyphenation}%
597  {*2ekernel | latexrelease}
598  \def\@outputpage{%
```

The `\endgroup` is put in by `\aftergroup`.

```
599  \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`.

```
600  \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 `verbatim` does not prevent hyphenation in the page head.

```
601  \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?

```
602  \@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.

```
603  \global\let\@if@newlist\if@newlist
604  \global@\newlistfalse
```

This next hook replaces the following:

```
\let\-\@dischyp
\let'\@acci\let`\@acci\let\=\@accii
\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\normalineskip  

\baselineskip\normalbaselineskip  

\sloppy  

605   \parboxrestore

```

... to here was in the command \@writesetup.

```

606   \shipout \vbox{%
607     \set@typeset@protect
608   \aftergroup \endgroup

```

Correct? or just restore by ending the group?

```

609   \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.

```

610   \if@specialpage
611     \global\@specialpagefalse\@nameuse{ps@\@specialstyle}%
612   \fi
613   \if@twoside
614     \ifodd\count\z@ \let\@thehead\@oddhead \let\@thefoot\@oddfoot
615       \let\@themargin\oddsidemargin
616     \else \let\@thehead\@evenhead
617       \let\@thefoot\@evenfoot \let\@themargin\evensidemargin
618     \fi
619   \fi

```

The rest was always inside the box.

RmS 91/08/15: added this line:

```

620   \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.

```

621   \normalsize

```

Reset the space factors.

```

622   \normalsfcodes

```

Reset these here (previously reset separately for head and foot)

```

623   \let\label\gobble
624   \let\index\gobble
625   \let\glossary\gobble
626   \baselineskip\z@skip \lineskip\z@skip \lineskiplimit\z@

```

... to here was in the command \shipoutsetup.

```
627   \begindvi
628   \vskip \topmargin
629   \moveright\@themargin \vbox {%
630     \setbox\@tempboxa \vbox to\headheight{%
631       \vfil
632       \color@hbox
633         \normalcolor
634         \hb@xt@\textwidth{\@thehead}%
635       \color@endbox
636     }%
637     \dp\@tempboxa \z@
638     \box\@tempboxa
639     \vskip \headsep
640     \box\@outputbox
641     \baselineskip \footskip
642     \color@hbox
643       \normalcolor
644       \hb@xt@\textwidth{\@thefoot}%
645     \color@endbox
646   }%
647 }%
```

\endgroup now inserted by \aftergroup

```
  Restore \if@newlist
648  \global\let\if@newlist\@@if@newlist
649  \global \colht \textheight
650  \stepcounter{page}%
651 }
```

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.

```
651 \let\firstmark\botmark
652 }
653 </2ekernel | latexrelease>
654 <latexrelease>\EndIncludeInRelease
655 <latexrelease>\IncludeInRelease{0000/00/00}%
656 <latexrelease> {\@outputpage}{Reset language for hyphenation}%
657 <latexrelease>\def\@outputpage{%
658 <latexrelease>\begingroup
659 <latexrelease> \let \protect \noexpand
660 <latexrelease> \resetactivechars
661 <latexrelease> \global\let\@@if@newlist\if@newlist
662 <latexrelease> \global\@newlistfalse
663 <latexrelease> \parboxrestore
664 <latexrelease> \shipout \vbox{%
665 <latexrelease>   \set@typeset@protect
666 <latexrelease>   \aftergroup \endgroup
667 <latexrelease>   \aftergroup \set@typeset@protect
668 <latexrelease>   \if@specialpage
669 <latexrelease>     \global\@specialpagefalse\@nameuse{ps@\@specialstyle}%
670 <latexrelease>   \fi
```

```

671 <{latexrelease}> \if@twoside
672 <{latexrelease}> \ifodd\count\z@
673 <{latexrelease}> \let\@thehead\@oddhead \let\@thefoot\@oddfoot
674 <{latexrelease}> \let\@themargin\oddsidemargin
675 <{latexrelease}> \else \let\@thehead\@evenhead
676 <{latexrelease}> \let\@thefoot\@evenfoot \let\@themargin\evensidemargin
677 <{latexrelease}> \fi
678 <{latexrelease}> \fi
679 <{latexrelease}> \reset@font
680 <{latexrelease}> \normalsize
681 <{latexrelease}> \normalsfcodes
682 <{latexrelease}> \let\label\@gobble
683 <{latexrelease}> \let\index\@gobble
684 <{latexrelease}> \let\glossary\@gobble
685 <{latexrelease}> \baselineskip\z@skip \lineskip\z@skip \lineskiplimit\z@
686 <{latexrelease}> \begindvi
687 <{latexrelease}> \vskip \topmargin
688 <{latexrelease}> \moveright\@themargin \vbox {%
689 <{latexrelease}> \setbox\@tempboxa \vbox to\headheight{%
690 <{latexrelease}> \vfil
691 <{latexrelease}> \color@hbox
692 <{latexrelease}> \normalcolor
693 <{latexrelease}> \hb@xt@\textwidth{\@thehead}%
694 <{latexrelease}> \color@endbox
695 <{latexrelease}> }%
696 <{latexrelease}> \dp\@tempboxa \z@
697 <{latexrelease}> \box\@tempboxa
698 <{latexrelease}> \vskip \headsep
699 <{latexrelease}> \box\@outputbox
700 <{latexrelease}> \baselineskip \footskip
701 <{latexrelease}> \color@hbox
702 <{latexrelease}> \normalcolor
703 <{latexrelease}> \hb@xt@\textwidth{\@thefoot}%
704 <{latexrelease}> \color@endbox
705 <{latexrelease}> }%
706 <{latexrelease}> }%
707 <{latexrelease}> \global\let\if@newlist\@@if@newlist
708 <{latexrelease}> \global \colht \textheight
709 <{latexrelease}> \stepcounter{page}%
710 <{latexrelease}> \let\firstmark\botmark
711 <{latexrelease}> }
712 <{latexrelease}> \EndIncludeInRelease
713 <{*2ekernel}>

```

(End of 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.

```

714 \def \begindvi{%
715   \unvbox \begindvibox
716   \global\let \begindvi \empty
717 }

```

(End of 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.

```

718 \def \@combinefloats {%
719   % \boxmaxdepth \maxdepth
720   \ifx \@toplist\@empty \else \@cflt \fi
721   \ifx \@botlist\@empty \else \@cflb \fi
722 }

723 \def \@cflt{%
724   \let \@elt \@comflelt
725   \setbox\@tempboxa \vbox{}%
726   \@toplist
727   \setbox\@outputbox \vbox{%
728     \boxmaxdepth \maxdepth
729     \unvbox\@tempboxa
730     \vskip -\floatsep
731     \topfigrule
732     \vskip \textfloatsep
733     \unvbox\@outputbox
734   }%
735   \let\@elt\relax
736   \xdef\@freelist{\@freelist\@toplist}%
737   \global\let\@toplist\@empty
738 }

739 \def \@cflb {%
740   \let\@elt\@comflelt
741   \setbox\@tempboxa \vbox{}%
742   \@botlist
743   \setbox\@outputbox \vbox{%
744     \unvbox\@outputbox
745     \vskip \textfloatsep
746     \botfigrule
747     \unvbox\@tempboxa
748     \vskip -\floatsep
749   }%
750   \let\@elt\relax
751   \xdef\@freelist{\@freelist\@botlist}%
752   \global \let \@botlist\@empty
753 }

```

(End of definition for \@combinefloats, \@cflt, and \@cflb.)

```

\@comflelt
\@comdblflflelt
754 \def\@comflelt#1{\setbox\@tempboxa
755   \vbox{\unvbox\@tempboxa\box #1\vskip\floatsep}}
756 \def\@comdblflflelt#1{\setbox\@tempboxa
757   \vbox{\unvbox\@tempboxa\box #1\vskip\dblfloatsep}}
758 \def \@combinedblfloats{%
759   \ifx \@dbltoplist \@empty
760   \else
761     \setbox\@tempboxa \vbox{}%
762     \let \@elt \@comdblflflelt

```

```

763     \@dbltoplist
764     \let \@elt \relax
765     \xdef \@freelist {\@freelist\@dbltoplist}%
766     \global \let \@dbltoplist \empty
767     \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.

```

768 {%\boxmaxdepth\maxdepth %% probably not needed, CAR
769 \unvbox\@tempboxa\vskip-\dblfloatsep

```

Here we need different typesetting if the top float comes from `\@topnewpage`.

```

770     \ifnum \@dbltopnum>\m@ne
771         \dblfigrule
772     \fi
773     \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.

```

774     \unvbox\@outputbox
775     }%
776     \fi
777 }
778 </2ekernel>

```

(End of definition for `\@comflelt`, `\@comdblfllelt`, 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`.

```

779 <*2ekernel | fltrace>
780 \def \@startcolumn {%
781     \global \@colroom \@colht
782     \@tryfcolumn \@deferlist
783     \if@fcolmade
784     (*trace)
785         \fl@trace{PAGE: float \if@twocolumn column \else page \fi
786                     completed}%
787     /trace>
788     \else
789     \begingroup
790         \let \reserved@b \@deferlist
791         \global \let \@deferlist \empty
792         \let \@elt \@scolelt
793         \reserved@b
794     \endgroup
795     \fi
796 }

```

This one does not need to set `\@colht`.

```
797 〈/2ekernel | fltrace〉  
798 〈latexrelease | fltrace〉\IncludeInRelease{2015/01/01}%">  
799 〈latexrelease | fltrace〉 {\@startdblcolumn}{float order in 2-column}%">  
800 〈*2ekernel | latexrelease | fltrace〉  
801 \def \@startdblcolumn {%802   \@tryfcolumn \@deferlist  
803   \if@fcolmade  
804   {fltrace} \fl@trace{PAGE: double float page completed}%">  
805   \else  
806     \begingroup  
807       \let \reserved@b \@deferlist  
808       \global \let \@deferlist \empty  
809       \let \@elt \@sdblcolelt  
810       \reserved@b  
811     \endgroup  
812   \fi  
813 }%  
814 〈/2ekernel | latexrelease | fltrace〉  
815 〈latexrelease | fltrace〉\EndIncludeInRelease  
816 〈latexrelease | fltrace〉\IncludeInRelease{0000/00/00}%">  
817 〈latexrelease | fltrace〉 {\@startdblcolumn}{float order in 2-column}%">  
818 〈latexrelease | fltrace〉\def \@startdblcolumn {%
```

Not needed since this always comes after `\@outputpage`:

```
819 〈latexrelease | fltrace〉% \global \@colht \textheight  
820 〈latexrelease | fltrace〉 \@tryfcolumn \@dbldeferlist  
821 〈latexrelease | fltrace〉 \if@fcolmade  
822 {*trace}  
823 〈latexrelease | fltrace〉 \fl@trace{PAGE: double float page completed}%">  
824 〈/trace〉  
825 〈latexrelease | fltrace〉 \else  
  
826 〈latexrelease | fltrace〉 \begingroup  
827 〈latexrelease | fltrace〉 \let \reserved@b \@dbldeferlist  
828 〈latexrelease | fltrace〉 \global \let \@dbldeferlist \empty  
829 〈latexrelease | fltrace〉 \let \@elt \@sdblcolelt  
830 〈latexrelease | fltrace〉 \reserved@b  
831 〈latexrelease | fltrace〉 \endgroup  
832 〈latexrelease | fltrace〉 \fi  
833 〈latexrelease | fltrace〉}%">  
834 〈latexrelease | fltrace〉\EndIncludeInRelease  
835 〈*2ekernel | fltrace〉
```

(End of definition for `\@startcolumn` and `\@startdblcolumn`.)

`\@tryfcolumn` Now tests if its list is empty before any further exertion.

```
836 \def \@tryfcolumn #1{  
837   \global \@fcolmadefalse  
838   \ifx #1\empty  
839   \else  
840   {*trace}  
841     \fl@trace{PAGE: try float \if@twocolumn column/page\else page\fi  
842           ---\string #1}%
```

```

843      \f1@trace{---- \string #1}%
844  
```

 $\langle/\text{trace}\rangle$

```

845      \xdef\@trylist{\#1}%
846      \global\let\@failedlist\empty
847      \begingroup
848      \let\@elt\@xtryfc\@trylist
849      \endgroup
850      \if@fcolmade
851      \v@tryfc \#1%
852      \fi
853  
```

 $\langle/\text{fi}\rangle$

```

854 }
855 
```

 $\langle/\text{2ekernel} | \text{ftrace}\rangle$

(End of definition for @tryfc .)

```

856 
```

 $\langle*\text{2ekernel}\rangle$

@scolelt

```

857 \def\@scolelt#1{\def\@currbox{\#1}\@addtonextcol}

```

(End of definition for @scolelt .)

@sdblcolelt

```

858 \def\@sdblcolelt#1{\def\@currbox{\#1}\@addtobblcol}

```

(End of definition for @sdblcolelt .)

@vtryfc

```

859 \def\@vtryfc #1{%
860   \global\setbox\@outputbox\vbox{}%
861   \let\@elt\@wtryfc
862   \f1succeed
863   \global\setbox\@outputbox\vbox{to\@colht}{%
864     \vskip\@fptop
865     \vskip-\@fpsep
866     \unvbox\@outputbox
867     \vskip\@fpbot}%
868   \let\@elt\relax
869   \xdef\#1{\@failedlist\f1fail}%
870   \xdef\@freelist{\@freelist\@f1succeed}}

```

(End of definition for @vtryfc .)

@wtryfc

```

871 \def\@wtryfc #1{%
872   \global\setbox\@outputbox\vbox{%
873     \unvbox\@outputbox
874     \vskip\@fpsep
875     \box\#1}}

```

(End of definition for @wtryfc .)

```

\@xtryfc

876 〈/2ekernel〉
877 〈\latexrelease〉\IncludeInRelease{2015/01/01}{\@xtryfc}%
878 〈\latexrelease〉
879 〈*2ekernel | \latexrelease〉
880 \def\@xtryfc #1{%
881   \@next\reserved@a\@trylist{}{}%
882   \@currtype \count #1%
883   \divide\@currtype\@xxxii
884   \multiply\@currtype\@xxxii
885   \bitor \@currtype \@failedlist
886   \testfp #1%
887   \@testwrongwidth #1%
888   \ifdim \ht #1>\@colht
889     \testtrue
890   \fi
891   \if@test
892     \cons\@failedlist #1%
893   \else
894     \@ytryfc #1%
895   \fi}%
896 〈/2ekernel | \latexrelease〉
897 〈\latexrelease〉\EndIncludeInRelease
898 〈\latexrelease〉\IncludeInRelease{0000/00/00}{\@xtryfc}%
899 〈\latexrelease〉
900 〈\latexrelease〉\def\@xtryfc #1{%
901 〈\latexrelease〉  \@next\reserved@a\@trylist{}{}%
902 〈\latexrelease〉  \@currtype \count #1%
903 〈\latexrelease〉  \divide\@currtype\@xxxii
904 〈\latexrelease〉  \multiply\@currtype\@xxxii
905 〈\latexrelease〉  \bitor \@currtype \@failedlist
906 〈\latexrelease〉  \testfp #1%
907 〈\latexrelease〉  \ifdim \ht #1>\@colht
908 〈\latexrelease〉    \testtrue
909 〈\latexrelease〉    \fi
910 〈\latexrelease〉    \if@test
911 〈\latexrelease〉      \cons\@failedlist #1%
912 〈\latexrelease〉    \else
913 〈\latexrelease〉      \@ytryfc #1%
914 〈\latexrelease〉    \fi}%
915 〈\latexrelease〉\EndIncludeInRelease
916 〈*2ekernel〉

```

(End of definition for \@xtryfc.)

```

\@ytryfc

917 \def\@ytryfc #1{%
918   \begingroup
919     \gdef\@flsucceed{\@elt #1}%
920     \global\let\@flfail\empty
921     \tempdima\ht #1%
922     \let\@elt\@ztryfc
923     \@trylist

```

```

924     \ifdim \@tempdima >\@fpmin
925         \global\@fcolmadetrue
926     \else
927         \@cons\@failedlist #1%
928     \fi
929 \endgroup
930 \if@fcolmade
931     \let\@elt\@gobble
932 \fi}

(End of definition for \ztryfc.)
```

\@ztryfc

```

933 〈/2ekernel〉
934 〈latexrelease〉\IncludeInRelease{2015/01/01}{\ztryfc}%
935 〈latexrelease〉
936 〈*2ekernel | latexrelease〉
937 \def\ztryfc #1{%
938     \@tempcnta\count #1%
939     \divide\@tempcnta\@xxxii
940     \multiply\@tempcnta\@xxxii
941     \@bitor \@tempcnta {\@failedlist \@flfail}%
942     \@testfp #1%
943     not in fixfloats?
944     \@testwrongwidth #1%
945     \@tempdimb\@tempdima
946     \advance\@tempdimb\ht #1%
947     \advance\@tempdimb\@fpsep
948     \ifdim \@tempdimb >\@colht
949         \@testtrue
950     \fi
951     \if@test
952         \@cons\@flfail #1%
953     \else
954         \@cons\@flsucceed #1%
955     \fi}%
956 〈/2ekernel | latexrelease〉
957 〈latexrelease〉\EndIncludeInRelease
958 〈latexrelease〉\IncludeInRelease{0000/00/00}{\ztryfc}%
959 〈latexrelease〉
960 〈latexrelease〉\def\ztryfc #1{%
961     \@tempcnta \count#1%
962     \divide\@tempcnta\@xxxii
963     \multiply\@tempcnta\@xxxii
964     \@bitor \@tempcnta {\@failedlist \@flfail}%
965     \@testfp #1%
966     \@tempdimb\@tempdima
967     \advance\@tempdimb\ht#1%
968     \advance\@tempdimb\@fpsep
969     \ifdim \@tempdimb >\@colht
970         \@testtrue
971     \fi

```

```

972 <|latexrelease> \if@test
973 <|latexrelease> \@cons\@flfail #1%
974 <|latexrelease> \else
975 <|latexrelease> \@cons\@flsucceed #1%
976 <|latexrelease> \@tempdima\@tempdimb
977 <|latexrelease> \fi}%
978 <|latexrelease>\EndIncludeInRelease

```

(End of 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.

```

979 <|*2ekernel | fltrace>
980 \def \@addtobot {%
981 <|*trace>
982 \f@trace{***Start addtobot}%
983 <|/trace>
984 \@getfpsbit 4\relax
985 <|*trace>
986 \f@trace{fpstype \ifodd \@tempcnta OK \else not \fi bot:
987 \the \@fpstype}%
988 <|/trace>
989 \ifodd \@tempcnta
990 \f@setnum \@botnum
991 \ifnum \@botnum>\z@
992 \tempswafalse
993 \f@checkspace \@botroom \@botlist
994 \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?

```

995 \global \maxdepth \z@
996 \f@updates \@botnum \@botroom \@botlist
997 <|*trace>
998 \f@trace{colroom (after-bot) = \the \@colroom}%
999 \f@trace{colnum (after-bot) = \the \@colnum}%
1000 \f@trace{botnum (after-bot) = \the \@botnum}%
1001 \f@trace{***Success: bot}%
1002 <|/trace>
1003 \qinserttrue
1004 \fi
1005 <|*trace>
1006 \else
1007 \f@trace{Fail: botnum = \the \@botnum:
1008 \fpstype \the \@fpstype=ORD?}%
1009 \ifnum \@fpstype<\sixt@n
1010 \f@trace{ERROR: !b float not successful (addtobot)}%
1011 \fi
1012 <|/trace>
1013 \fi
1014 \fi
1015 }

```

(End of definition for \@addtobot.)

\@addtotoporbot Lots of changes.

```
1016 \def \@addtotoporbot {%
1017   <*trace>
1018     \f1@trace{***Start addtotoporbot}%
1019   </trace>
1020     \@getfpsbit \tw@
1021   <*trace>
1022     \f1@trace{fpstype \ifodd \@tempcnta OK \else not \fi top:
1023                                         \the \@fpstype}%
1024   </trace>
1025     \ifodd \@tempcnta
1026       \@flsetnum \@topnum
1027       \ifnum \@topnum>\z@
1028         \@tempswafalse
1029         \@flcheckspace \@toproom \@toplist
1030         \if@tempswa
1031           \obitor\@currtype{\@midlist\@botlist}%
1032   <*trace>
1033     \f1@trace{(mid+bot)list: \@midlist, \@botlist:
1034                           (addtotoporbot-before)}%
1035   </trace>
1036     \if@test
1037   <*trace>
1038     \f1@trace{type already on list: mid or bot---sent to addtobot}%
1039   </trace>
1040   \else
1041     \@flupdates \@topnum \@toproom \@toplist
1042   <*trace>
1043     \f1@trace{colroom (after-top) = \the \@colroom}%
1044     \f1@trace{colnum (after-top) = \the \@colnum}%
1045     \f1@trace{topnum (after-top) = \the \@topnum}%
1046     \f1@trace{***Success: top}%
1047   </trace>
1048     \@inserttrue
1049   \fi
1050   \fi
1051   <*trace>
1052   \else
1053     \f1@trace{Fail: topnum = \the \@topnum: fpstype
1054                                         \the \@fpstype=ORD?}%
1055     \ifnum \@fpstype<\sixt@n
1056       \f1@trace{ERROR: !t float not successful (addtotoporbot)}%
1057     \fi
1058   </trace>
1059   \fi
1060   \fi
1061   \if@insert
1062   \else
1063   <*trace>
1064     \f1@trace{sent to addtobot (addtotoporbot)}%
1065   </trace>
1066   \@addtobot
1067   \fi
1068 }
```

```

1069  ⟨/2ekernel | fltrace⟩

(End of definition for \@addtotoporbot.)
```

\@addtocurcol Lots of changes.

```

1070  ⟨| latexrelease | fltrace | flafter⟩\IncludeInRelease{2015/01/01}%
1071  ⟨| latexrelease | fltrace | flafter⟩ {⟨\@addtocurcol⟩{float order in 2-column}%
1072  ⟨*2ekernel | latexrelease | fltrace | flafter⟩
1073  \def \@addtocurcol {%
1074  ⟨*trace⟩
1075    \fl@trace{***Start addtocurcol}%
1076  ⟨/trace⟩
1077    \cinsertfalse
1078    \csetfloattypecounts
1079    \ifnum \cfpstype=8
1080    ⟨*trace⟩
1081      \fl@trace{fpstype !p only (addtocurcol): \the \cfpstype = 8?}%
1082  ⟨/trace⟩
1083    \else
1084      \ifnum \cfpstype=24
1085    ⟨*trace⟩
1086      \fl@trace{fpstype p only (addtocurcol): \the \cfpstype = 24?}%
1087  ⟨/trace⟩
1088    \else
1089      \cflsettextmin
```

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.

```

1090  ⟨*trace⟩
1091    \fl@trace{textfloatsheight (before) = \the \textfloatsheight}%
1092  ⟨/trace⟩
1093    \advance \textmin \textfloatsheight
1094    \reqcolroom \pageht
```

This line must be removed since \specialoutput changed.

```

1095 %       \advance \reqcolroom \pagedp
1096 ⟨*trace⟩
1097   \fl@trace{textmin + textfloatsheight: \the \textmin}%
1098   \fl@trace{page-so-far: \the \reqcolroom}%
1099 ⟨/trace⟩
1100   \ifdim \textmin>\reqcolroom
1101     \reqcolroom \textmin
1102 ⟨*trace⟩
1103   \fl@trace{ORD? textmin being used}%
1104 ⟨/trace⟩
1105   \fi
1106   \advance \reqcolroom \ht\currbox
1107 ⟨*trace⟩
1108   \fl@trace{float size = \the \ht \currbox (addtocurcol)}%
1109   \fl@trace{colroom = \the \colroom (addtocurcol)}%
1110   \fl@trace{reqcolroom = \the \reqcolroom (addtocurcol)}%
1111 ⟨/trace⟩
```

```

1112      \ifdim \@colroom>\@reqcolroom
1113          \@flsetnum \@colnum
1114          \ifnum \@colnum>\z@
1115              \@bitor\@currtype\@deferlist

```

We need to defer the float also if its width doesn't fit.

```

1116          \@testwrongwidth\@currbox
1117  <*trace>
1118      \fl@trace{deferlist: \@deferlist: (addtocurcol-before)}%
1119  </trace>
1120      \if@test
1121  <*trace>
1122      \fl@trace{type already on list: defer (addtocurcol)}%
1123  </trace>
1124      \else
1125          \@bitor\@currtype\@botlist
1126  <*trace>
1127      \fl@trace{botlist: \@botlist: (addtocurcol-before)}%
1128  </trace>
1129      \if@test
1130  <*trace>
1131      \fl@trace{type already on list: bot---sent to addtobot}%
1132  </trace>
1133      \@addtobot
1134      \else
1135  <*trace>
1136      \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi
1137          here: \the \@fpstype}%
1138  </trace>
1139      \ifodd \count\@currbox
1140          \advance \@reqcolroom \intextsep
1141          \ifdim \@colroom>\@reqcolroom
1142              \global \advance \@colnum \m@ne
1143              \global \advance \@textfloatsheight \ht\@currbox

```

This may sometimes give an overestimate.

```

1144          \global \advance \@textfloatsheight 2\intextsep
1145          \@cons \@midlist \@currbox
1146  <*trace>
1147      \fl@trace{***Success: here}%
1148      \fl@trace{textfloatsheight (after-here) =
1149          \the \@textfloatsheight}%
1150      \fl@trace{colnum (after-here) = \the \@colnum}%
1151  </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.
 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).

```

1152          \if@nobreak
1153              \nobreak
1154              \nobreakfalse
1155              \everypar{}%
1156          \else
1157              \addpenalty \interlinepenalty
1158          \fi
1159          \vskip \intextsep
1160          \box@\currbox
1161          \penalty\interlinepenalty
1162          \vskip\intextsep
1163          \ifnum\outputpenalty <- \Mii \vskip -\parskip\fi

```

Typesetting ends here.

```

1164          \outputpenalty \z@
1165          \inserttrue
1166      {*trace}
1167      \else
1168          \f@trace{Fail---no room at 2nd test of colroom
1169                      (addtocorcol \string\intextsep)}%
1170  
```

 $\{/trace\}$

```

1171          \fi
1172          \fi
1173          \if@insert
1174          \else

```

Next set of docstrip guards are a bit weird, essentially \c@addtotopbot ends up inside the kernel and the `ftrace` package and \c@addtobot shows up in the `flafter` package. Guess that could have been done a bit more obvious :-)

```

1175  {*2ekernel | ftrace | latexrelease}
1176  {*trace}
1177      \f@trace{not here: sent to addtotopbot}%
1178  
```

 $\{/trace\}$

```

1179          \c@addtotopbot
1180  
```

 $\{/2ekernel | ftrace | latexrelease\}$

```

1181  {*}2ekernel&!ftrace&!latexrelease}
1182  {*trace}
1183      \f@trace{not here: sent to addtobot}%
1184  
```

 $\{/trace\}$

```

1185          \c@addtobot
1186  
```

 $\{/!2ekernel&!ftrace&!latexrelease\}$

```

1187          \fi
1188          \fi
1189          \fi
1190  
```

 $\{*trace\}$

```

1191          \else
1192              \f@trace{Fail: colnum = \the \c@colnum:
1193                          fpstype \the \c@fpstype=ORD?}%
1194              \ifnum \c@fpstype<\sixt@n
1195                  \f@trace{ERROR: BANG float not successful (addtocurcol)}%
1196              \fi

```

```

1197 〈/trace〉
1198          \fi
1199 〈*trace〉
1200          \else
1201          \fl@trace{Fail---no room: fl box ht: \the \ht \currbox
1202                                         (addtocurcol)}%
1203 〈/trace〉
1204          \fi
1205          \fi
1206          \fi
1207          \if@insert
1208          \else
1209          \@resethfps
1210 〈*trace〉
1211          \fl@trace{put on deferlist (addtocurcol)}%
1212 〈/trace〉
1213          \@cons\@deferlist\@currbox
1214 〈*trace〉
1215          \fl@trace{deferlist: \@deferlist: (addtocurcol-after)}%
1216 〈/trace〉
1217          \fi
1218 }%
1219 (./2ekernel | latexrelease | fltrace | flafter)
1220 〈latexrelease | fltrace | flafter〉\EndIncludeInRelease
1221 〈latexrelease | fltrace | flafter〉\IncludeInRelease{0000/00/00}%
1222 〈latexrelease | fltrace | flafter〉 {\@addtocurcol}{float order in 2-column}%
1223 〈latexrelease | fltrace | flafter〉\def \@addtocurcol {%
1224 〈*trace〉
1225 〈latexrelease | fltrace | flafter〉 \fl@trace{***Start addtocurcol}%
1226 〈/trace〉
1227 〈latexrelease | fltrace | flafter〉 \@insertfalse
1228 〈latexrelease | fltrace | flafter〉 \@setfloattypecounts
1229 〈latexrelease | fltrace | flafter〉 \ifnum \@fpstype=8
1230 〈*trace〉
1231 〈latexrelease | fltrace | flafter〉 \fl@trace{fpstype !p only (addtocurcol):
1232 〈latexrelease | fltrace | flafter〉 \the \@fpstype = 8?}%
1233 〈/trace〉
1234 〈latexrelease | fltrace | flafter〉 \else
1235 〈latexrelease | fltrace | flafter〉 \ifnum \@fpstype=24
1236 〈*trace〉
1237 〈latexrelease | fltrace | flafter〉 \fl@trace{fpstype p only (addtocurcol):
1238 〈latexrelease | fltrace | flafter〉 \the \@fpstype = 24?}%
1239 〈/trace〉
1240 〈latexrelease | fltrace | flafter〉 \else
1241 〈latexrelease | fltrace | flafter〉 \@flesstextmin

```

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.

```

1242 〈*trace〉
1243 〈latexrelease | fltrace | flafter〉 \fl@trace{textfloatsheight (before) =
1244 〈latexrelease | fltrace | flafter〉 \the \@textfloatsheight}%
1245 〈/trace〉

```

```

1246 〈latexrelease | fltrace | flafter〉      \advance \@textmin \@textfloatsheight
1247 〈latexrelease | fltrace | flafter〉      \@reqcolroom \@pageht
This line must be removed since \@specialoutput changed.
1248 %           \advance \@reqcolroom \@pagedp
1249 〈*trace〉
1250 〈latexrelease | fltrace | flafter〉      \fl@trace{textmin + textfloatsheight:
1251 〈latexrelease | fltrace | flafter〉          \the \@textmin}%
1252 〈latexrelease | fltrace | flafter〉      \fl@trace{page-so-far: \the \@reqcolroom}%
1253 〈latexrelease | fltrace | flafter〉
1254 〈/trace〉
1255 〈latexrelease | fltrace | flafter〉
1256 〈latexrelease | fltrace | flafter〉
1257 〈*trace〉
1258 〈latexrelease | fltrace | flafter〉
1259 〈/trace〉
1260 〈latexrelease | fltrace | flafter〉
1261 〈latexrelease | fltrace | flafter〉
1262 〈*trace〉
1263 〈latexrelease | fltrace | flafter〉
1264 〈latexrelease | fltrace | flafter〉
1265 〈latexrelease | fltrace | flafter〉
1266 〈latexrelease | fltrace | flafter〉
1267 〈latexrelease | fltrace | flafter〉
1268 〈latexrelease | fltrace | flafter〉
1269 〈/trace〉
1270 〈latexrelease | fltrace | flafter〉
1271 〈latexrelease | fltrace | flafter〉
1272 〈latexrelease | fltrace | flafter〉
1273 〈latexrelease | fltrace | flafter〉
1274 〈*trace〉
1275 〈latexrelease | fltrace | flafter〉
1276 〈latexrelease | fltrace | flafter〉
1277 〈/trace〉
1278 〈latexrelease | fltrace | flafter〉
1279 〈*trace〉
1280 〈latexrelease | fltrace | flafter〉
1281 〈latexrelease | fltrace | flafter〉
1282 〈/trace〉
1283 〈latexrelease | fltrace | flafter〉
1284 〈latexrelease | fltrace | flafter〉
1285 〈*trace〉
1286 〈latexrelease | fltrace | flafter〉
1287 〈latexrelease | fltrace | flafter〉
1288 〈/trace〉
1289 〈latexrelease | fltrace | flafter〉
1290 〈*trace〉
1291 〈latexrelease | fltrace | flafter〉
1292 〈latexrelease | fltrace | flafter〉
1293 〈/trace〉
1294 〈latexrelease | fltrace | flafter〉
1295 〈latexrelease | fltrace | flafter〉
1296 〈*trace〉
1297 〈latexrelease | fltrace | flafter〉
1298 〈latexrelease | fltrace | flafter〉      \advance \@textmin \@textfloatsheight
                                              \@reqcolroom \@pageht
\ifdim \@textmin>\@reqcolroom
  \@reqcolroom \@textmin
    \fl@trace{ORD? textmin being used}%
\fi
\advance \@reqcolroom \ht\@currbox
\fl@trace{float size =
  \the \ht \@currbox (addtocurcol)}%
\fl@trace{colroom =
  \the \@colroom (addtocurcol)}%
\fl@trace{reqcolroom =
  \the \@reqcolroom (addtocurcol)}%
\ifdim \@colroom>\@reqcolroom
  \@flsetnum \@colnum
  \ifnum \@colnum>\z@
    \@bitor\@currtype\@deferlist
    \fl@trace{deferlist:
      \@deferlist: (addtocurcol-before)}%
  \if@test
    \fl@trace{type already on list:
      defer (addtocurcol)}%
  \else
    \@bitor\@currtype\@botlist
    \fl@trace{botlist: \@botlist:
      (addtocurcol-before)}%
  \if@test
    \fl@trace{type already on list:
      bot---sent to addtobot}%
    \@addtobot
  \else
    \fl@trace{fpstype
      \ifodd \@tempcnta OK \else not \fi

```

```

1299 <{latexrelease | fltrace | flafter}>
1300 </trace>
1301 <{latexrelease | fltrace | flafter}>
1302 <{latexrelease | fltrace | flafter}>
1303 <{latexrelease | fltrace | flafter}>
1304 <{latexrelease | fltrace | flafter}>
1305 <{latexrelease | fltrace | flafter}>
1306 <{latexrelease | fltrace | flafter}>
1307 <{latexrelease | fltrace | flafter}>
1308 <{latexrelease | fltrace | flafter}>
1309 <{latexrelease | fltrace | flafter}>
1310 <{*trace}>
1311 <{latexrelease | fltrace | flafter}>
1312 <{latexrelease | fltrace | flafter}>
1313 <{latexrelease | fltrace | flafter}>
1314 <{latexrelease | fltrace | flafter}>
1315 <{latexrelease | fltrace | flafter}>
1316 <{latexrelease | fltrace | flafter}>
1317 </trace>

This may sometimes give an overestimate.

CHANGE TO \addtocurcol:
\penalty\z@ changed to \penalty\interlinepenalty so \samepage works prop-
erly with figure and table environments. (Changed 23 Oct 86)
There is also an \addpenalty\interlinepenalty above.
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).

1318 <{latexrelease | fltrace | flafter}>
1319 <{latexrelease | fltrace | flafter}>
1320 <{latexrelease | fltrace | flafter}>
1321 <{latexrelease | fltrace | flafter}>
1322 <{latexrelease | fltrace | flafter}>
1323 <{latexrelease | fltrace | flafter}>
1324 <{latexrelease | fltrace | flafter}>
1325 <{latexrelease | fltrace | flafter}>
1326 <{latexrelease | fltrace | flafter}>
1327 <{latexrelease | fltrace | flafter}>
1328 <{latexrelease | fltrace | flafter}>
1329 <{latexrelease | fltrace | flafter}>
1330 <{latexrelease | fltrace | flafter}>
1331 <{latexrelease | fltrace | flafter}>

Typesetting ends here.

1332 <{latexrelease | fltrace | flafter}>
1333 <{latexrelease | fltrace | flafter}>
1334 <{*trace}>
1335 <{latexrelease | fltrace | flafter}>
```

```

here: \the \cftpstype}%
\ifodd \count\currbox
\advance \reqcolroom \intextsep
\ifdim \colroom\reqcolroom
\global \advance \colnum \m@ne
\global \advance
\textfloatsheight\ht\currbox

\global \advance
\textfloatsheight 2\intextsep
\cons \midlist \currbox

\f@trace{***Success: here}%
\f@trace{textfloatsheight
(after-here) =
\the \textfloatsheight}%
\f@trace{colnum (after-here) =
\the \colnum}%
```

```

1336 <{latexrelease | fltrace | flafter}> \fl@trace{Fail---no room at 2nd test of colroom
1337 <{latexrelease | fltrace | flafter}> (addtocorcol \string\intextsep)}%
1338 </trace>
1339 <{latexrelease | fltrace | flafter}> \fi
1340 <{latexrelease | fltrace | flafter}> \fi
1341 <{latexrelease | fltrace | flafter}> \if@insert
1342 <{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 :-)

```

1343 <{*2ekernel | fltrace}>
1344 <{*trace}>
1345 <{latexrelease | fltrace | flafter}> \fl@trace{not here: sent to addtotoporbot}%
1346 </trace>
1347 <{latexrelease | fltrace | flafter}> \@addtotoporbot
1348 </2ekernel | fltrace>
1349 <{*!2ekernel&!autoload&!fltrace}>
1350 <{*trace}>
1351 <{latexrelease | fltrace | flafter}> \fl@trace{not here: sent to addtobot}%
1352 </trace>
1353 <{latexrelease | fltrace | flafter}> \@addtobot
1354 </!2ekernel&!autoload&!fltrace>
1355 <{latexrelease | fltrace | flafter}> \fi
1356 <{latexrelease | fltrace | flafter}> \fi
1357 <{latexrelease | fltrace | flafter}> \fi
1358 <{*trace}>
1359 <{latexrelease | fltrace | flafter}> \else
1360 <{latexrelease | fltrace | flafter}> \fl@trace{Fail: colnum = \the \@colnum:
1361 <{latexrelease | fltrace | flafter}> fpstype \the \@fpstype=ORD?}%
1362 <{latexrelease | fltrace | flafter}> \ifnum \@fpstype<\sixt@n
1363 <{latexrelease | fltrace | flafter}> \fl@trace{ERROR: BANG float not successful
1364 <{latexrelease | fltrace | flafter}> (addtocurcol)}%
1365 <{latexrelease | fltrace | flafter}> \fi
1366 </trace>
1367 <{latexrelease | fltrace | flafter}> \fi
1368 <{*trace}>
1369 <{latexrelease | fltrace | flafter}> \else
1370 <{latexrelease | fltrace | flafter}> \fl@trace{Fail---no room: fl box ht:
1371 <{latexrelease | fltrace | flafter}> \the \ht \@currbox (addtocurcol)}%
1372 </trace>
1373 <{latexrelease | fltrace | flafter}> \fi
1374 <{latexrelease | fltrace | flafter}> \fi
1375 <{latexrelease | fltrace | flafter}> \fi
1376 <{latexrelease | fltrace | flafter}> \if@insert
1377 <{latexrelease | fltrace | flafter}> \else
1378 <{latexrelease | fltrace | flafter}> \resethfps
1379 <{*trace}>
1380 <{latexrelease | fltrace | flafter}> \fl@trace{put on deferlist (addtocurcol)}%
1381 </trace>
1382 <{latexrelease | fltrace | flafter}> \@cons\@deferlist\@currbox
1383 <{*trace}>
1384 <{latexrelease | fltrace | flafter}> \fl@trace{deferlist: \@deferlist:
1385 <{latexrelease | fltrace | flafter}> (addtocurcol-after)}%

```

```

1386  </trace>
1387  <|latexrelease | fltrace | flafter>    \fi
1388  <|latexrelease | fltrace | flafter>  }%
1389  <|latexrelease | fltrace | flafter>\EndIncludeInRelease

```

(End of definition for \@addtocurcol.)

\@addtonextcol Lots of changes.

```

1390  <|latexrelease | fltrace>\IncludeInRelease{2015/01/01}
1391  <|latexrelease | fltrace>  {\@addtonextcol}{float order in 2-column}%
1392  <*2ekernel | latexrelease | fltrace>
1393  \def\@addtonextcol{%
1394      \begingroup
1395      <*trace>
1396          \fl@trace{***Start addtonextcol}%
1397      </trace>
1398      \insertfalse
1399      \setfloattypecounts
1400      \ifnum \fpstype=8
1401      <*trace>
1402          \fl@trace{fpstype not curcol: \the \fpstype = 8?}%
1403      </trace>
1404      \else
1405          \ifnum \fpstype=24
1406      <*trace>
1407          \fl@trace{fpstype not curcol: \the \fpstype = 24?}%
1408      </trace>
1409      \else
1410          \flsettextmin
1411      <*trace>
1412          \fl@trace{text-so-far: Opt (top of col)}%
1413      </trace>
1414      \reqcolroom \ht\currbox
1415      <*trace>
1416          \fl@trace{float size: \the \reqcolroom (addtonextcol)}%
1417      </trace>
1418      \advance \reqcolroom \textmin
1419      <*trace>
1420          \fl@trace{colroom = \the \colroom (addtonextcol)}%
1421          \fl@trace{reqcolroom = \the \reqcolroom (addtonextcol)}%
1422      </trace>
1423          \ifdim \colroom>\reqcolroom
1424              \flsetnum \colnum
1425              \ifnum\colnum>\z@
1426                  \bitor\currtype\deferlist
1427              <*trace>
1428                  \fl@trace{deferlist: \deferlist: (addtonextcol-before)}%
1429      </trace>
1430          \testwidth\currbox
1431          \if@test
1432              <*trace>
1433                  \fl@trace{type already on list: defer (addtonextcol)}%
1434      </trace>

```

```

1435           \else
1436   <*trace>
1437           \fl@trace{sent to addtotoporbot (addtonextcol)}%
1438   </trace>
1439           \@addtotoporbot
1440           \fi
1441   <*trace>
1442   \else
1443       \fl@trace{Fail---no room: fl box ht: \the \ht \currbox
1444                                         (addtonextcol)}%
1445   </trace>
1446   \fi
1447   \fi
1448   \fi
1449   \if@insert
1450   \else
1451   <*trace>
1452       \fl@trace{put back on deferlist (addtonextcol)}%
1453   </trace>
1454       \@cons\@deferlist\@currbox
1455   <*trace>
1456       \fl@trace{deferlist: \@deferlist: (addtonextcol-after)}%
1457   </trace>
1458   \fi
1459   <*trace>
1460   \fl@trace{End of addtonextcol -- locally counts:}%
1461   \fl@trace{col: \the\colnum. top: \the \topnum. bot: \the \botnum.}%
1462   </trace>
1463   \endgroup
1464   <*trace>
1465   \fl@trace{End of addtonextcol -- globally counts:}%
1466   \fl@trace{col: \the\colnum. top: \the \topnum. bot: \the \botnum.}%
1467   </trace>
1468   }%
1469 }%
1470 </2ekernel | latexrelease | fltrace>
1471 <latexrelease | fltrace>\EndIncludeInRelease
1472 <latexrelease | fltrace>\IncludeInRelease{0000/00/00}%
1473 <latexrelease | fltrace> {\@addtonextcol}{float order in 2-column}%
1474 <latexrelease | fltrace>\def\@addtonextcol{%
1475 <latexrelease | fltrace> \begingroup
1476 <*trace>
1477 <latexrelease | fltrace> \fl@trace{***Start addtonextcol}%
1478 </trace>
1479 <latexrelease | fltrace> \@insertfalse
1480 <latexrelease | fltrace> \@setfloattypecounts
1481 <latexrelease | fltrace> \ifnum \fpstype=8
1482 <*trace>
1483 <latexrelease | fltrace> \fl@trace{fpstype not curcol:
1484 <latexrelease | fltrace> \the \fpstype = 8?}%
1485 </trace>
1486 <latexrelease | fltrace> \else
1487 <latexrelease | fltrace> \ifnum \fpstype=24
1488 <*trace>

```

```

1489 <{latexrelease | fltrace> \f1@trace{fpstype not curcol:
1490 <{latexrelease | fltrace> \the \fpstype = 24?}%
1491 </trace>
1492 <{latexrelease | fltrace> \else
1493 <{latexrelease | fltrace> \f1@trace{text-so-far: Opt (top of col)}%
1494 <{*trace}> \reqcolroom \ht\currbox
1495 <{latexrelease | fltrace> \f1@trace{float size:
1496 </trace> \the \reqcolroom (addtonextcol)}%
1497 <{latexrelease | fltrace> \reqcolroom \ht\currbox
1498 <{*trace}> \f1@trace{float size:
1499 <{latexrelease | fltrace> \the \reqcolroom (addtonextcol)}%
1500 <{latexrelease | fltrace> \reqcolroom \ht\currbox
1501 <{latexrelease | fltrace> \f1@trace{float size:
1502 </trace> \the \reqcolroom \ht\currbox
1503 <{latexrelease | fltrace> \reqcolroom \ht\currbox
1504 <{*trace}> \f1@trace{colroom =
1505 <{latexrelease | fltrace> \the \colroom (addtonextcol)}%
1506 <{latexrelease | fltrace> \f1@trace{reqcolroom =
1507 <{latexrelease | fltrace> \the \reqcolroom (addtonextcol)}%
1508 <{latexrelease | fltrace> \f1@trace{reqcolroom =
1509 </trace> \ifdim \colroom>\reqcolroom
1510 <{latexrelease | fltrace> \f1@trace{colroom =
1511 <{latexrelease | fltrace> \the \colroom (addtonextcol)}%
1512 <{latexrelease | fltrace> \f1@trace{reqcolroom =
1513 <{latexrelease | fltrace> \the \reqcolroom (addtonextcol)}%
1514 <{*trace}> \f1@trace{deferlist: \deferlist:
1515 <{latexrelease | fltrace> \addtonextcol-before)}%
1516 <{latexrelease | fltrace> \f1@trace{type already on list:
1517 </trace> \addtonextcol)}%
1518 <{latexrelease | fltrace> \f1@trace{type already on list:
1519 <{*trace}> \addtonextcol)}%
1520 <{latexrelease | fltrace> \f1@trace{sent to addtotopbot
1521 <{latexrelease | fltrace> \addtonextcol)}%
1522 </trace> \f1@trace{sent to addtotopbot
1523 <{latexrelease | fltrace> \addtonextcol)}%
1524 <{*trace}> \f1@trace{sent to addtotopbot
1525 <{latexrelease | fltrace> \addtonextcol)}%
1526 <{latexrelease | fltrace> \f1@trace{sent to addtotopbot
1527 </trace> \addtonextcol)}%
1528 <{latexrelease | fltrace> \f1@trace{sent to addtotopbot
1529 <{latexrelease | fltrace> \addtonextcol)}%
1530 <{latexrelease | fltrace> \f1@trace{sent to addtotopbot
1531 <{*trace}> \addtonextcol)}%
1532 <{latexrelease | fltrace> \f1@trace{Fail---no room: f1 box ht:
1533 <{latexrelease | fltrace> \the \ht\currbox (addtonextcol)}%
1534 <{latexrelease | fltrace> \f1@trace{Fail---no room: f1 box ht:
1535 </trace> \addtonextcol)}%
1536 <{latexrelease | fltrace> \f1@trace{Fail---no room: f1 box ht:
1537 <{latexrelease | fltrace> \addtonextcol)}%
1538 <{latexrelease | fltrace> \f1@trace{Fail---no room: f1 box ht:
1539 <{latexrelease | fltrace> \addtonextcol)}%
1540 <{latexrelease | fltrace> \f1@trace{put back on deferlist
1541 <{*trace}>
1542 <{latexrelease | fltrace>
```

```

1543 <{latexrelease | fltrace}> (addtonextcol)%  

1544 </trace>  

1545 <{latexrelease | fltrace}> \@cons\@deferlist\@currbox  

1546 <*trace>  

1547 <{latexrelease | fltrace}> \fl@trace{\deferlist: \@deferlist:  

1548 <{latexrelease | fltrace}> (addtonextcol-after)}%  

1549 </trace>  

1550 <{latexrelease | fltrace}> \fi  

1551 <*trace>  

1552 <{latexrelease | fltrace}> \fl@trace{End of addtonextcol --  

1553 <{latexrelease | fltrace}> locally counts:}%  

1554 <{latexrelease | fltrace}> \fl@trace{col: \the \@colnum.  

1555 <{latexrelease | fltrace}> top: \the \@topnum. bot: \the \@botnum.}%  

1556 </trace>  

1557 <{latexrelease | fltrace}> \endgroup  

1558 <*trace>  

1559 <{latexrelease | fltrace}> \fl@trace{End of addtonextcol --  

1560 <{latexrelease | fltrace}> globally counts:}%  

1561 <{latexrelease | fltrace}> \fl@trace{col: \the \@colnum.  

1562 <{latexrelease | fltrace}> top: \the \@topnum. bot: \the \@botnum.}%  

1563 </trace>  

1564 <{latexrelease | fltrace}>%  

1565 <{latexrelease | fltrace}>\EndIncludeInRelease

```

(End of definition for \@addtonextcol.)

\@addtoblcol Lots of changes.

```

1566 <{latexrelease | fltrace}>\IncludeInRelease{2015/01/01}%  

1567 <{latexrelease | fltrace}> {\@addtoblcol}{float order in 2-column}%">  

1568 <{2ekernel | latexrelease | fltrace}>  

1569 \def\@addtoblcol{  

1570   \begingroup  

1571   <*trace>  

1572   \fl@trace{***Start addtoblcol}%">  

1573 </trace>  

1574   \@insertfalse  

1575   \@setfloattypecounts  

1576   \@getfpsbit \tw@  

1577   <*trace>  

1578   \fl@trace{fpstype \ifodd \tempcnta OK \else not \fi dbltop:  

1579   \the \@fpstype}%">  

1580 </trace>  

1581   \ifodd\@tempcnta  

1582     \@flsetnum \dbltopnum  

1583     \ifnum \dbltopnum>\z@  

1584       \@tempswafalse  

1585       \ifdim \dbltoproom>\ht\@currbox  

1586         \@tempswatrue  

1587       <*trace>  

1588       \fl@trace{Space OK: \dbltoproom =  

1589       \the \dbltoproom > \the \ht \@currbox  

1590       (dbltoproom)}%  

1591     </trace>  

1592     \else

```

```

1593  <*trace>
1594    \fl@trace{fpstype: \the \fpstype (addtoblcol)}%
1595  </trace>
1596  \ifnum \fpstype<\sixt@n
1597  <*trace>
1598    \fl@trace{BANG float ignoring \@dbltoproom}%
1599    \fl@trace{\@spaces \@dbltoproom = \the \@dbltoproom.
1600              Ht float: \the \ht \currbox-BANG}%
1601 </trace>

```

Need to check that there is room on the page, using the local value of \textmin to make the necessary adjustment to \@dbltoproom.

```

1602          \advance \@dbltoproom \textmin
1603  <*trace>
1604    \fl@trace{Local value of texmin: \the\textmin}%
1605    \fl@trace{\@spaces space on page = \the \@dbltoproom.
1606              Ht float: \the \ht \currbox-BANG}%
1607 </trace>
1608  \ifdim \@dbltoproom>\ht\currbox
1609    \tempswatru
1610  <*trace>
1611    \fl@trace{Space OK BANG: space on page =
1612              \the \@dbltoproom > \the \ht \currbox}%
1613  \else
1614    \fl@trace{fpstype: \the \fpstype}%
1615    \fl@trace{Fail---no room dbltoproom-BANG?:}%
1616    \fl@trace{\@spaces space on page = \the \@dbltoproom.
1617              Ht float: \the \ht \currbox}%
1618 </trace>
1619  \fi
1620  \advance \@dbltoproom -\textmin
1621  <*trace>
1622  \else
1623    \fl@trace{fpstype: \the \fpstype}%
1624    \fl@trace{Fail---no room dbltoproom-ORD?:}%
1625    \fl@trace{\@spaces \@dbltoproom = \the \@dbltoproom.
1626              Ht float: \the \ht \currbox}%
1627 </trace>
1628  \fi
1629  \fi
1630  \if@tempswa
1631    \bitor \currtype \deferlist
1632  <*trace>
1633    \fl@trace{(dbl)deferlist: \deferlist: (before)}%
1634 </trace>
      not in fixfloats?
1635          \testwidth\currbox
1636          \if@test
1637  <*trace>
1638    \fl@trace{type already on list: (dbl)defer}%
1639 </trace>
1640  \else
1641          \tempdima -\ht\currbox

```

```

1642           \advance\@tempdima
1643             -\ifx \dbltoplist\empty \dbltxtfloatsep \else
1644               \dblfloatsep \fi
1645             \global \advance \dbltoproom \@tempdima
1646             \global \advance \colht \@tempdima
1647             \global \advance \dbltopnum \m@ne
1648             \cons \dbltoplist \currbox
1649   <*trace>
1650     \f@trace{dbltopnum (after) = \the \dbltopnum}%
1651     \f@trace{***Success: dbltop}%
1652   </trace>
1653     \cinserttrue
1654   \fi
1655   \fi
1656   <*trace>
1657   \else
1658     \f@trace{Fail: dbltopnum = \the \dbltopnum: fpstype
1659                           \the \fpstype=ORD?}%
1660     \ifnum \fpstype<\sixt@n
1661       \f@trace{ERROR: !t float not successful (addtoblcol)}%
1662     \fi
1663   </trace>
1664   \fi
1665   \fi
1666   \cinsert
1667   \else
1668   <*trace>
1669     \f@trace{put on deferlist}%
1670   </trace>
1671   \cons\@deferlist\currbox
1672   <*trace>
1673     \f@trace{(dbl)deferlist: \@deferlist: (after)}%
1674   </trace>
1675   \fi
1676   <*trace>
1677     \f@trace{End of addtoblcol -- locally count:}%
1678     \f@trace{ dbltop: \the \dbltopnum.}%
1679   </trace>
1680   \endgroup
1681   <*trace>
1682     \f@trace{End of addtoblcol -- globally count:}%
1683     \f@trace{dbltop: \the \dbltopnum.}%
1684   </trace>
1685 }%
1686 </2ekernel | latexrelease | fltrace>
1687 <(latexrelease | fltrace)\EndIncludeInRelease
1688 <(latexrelease | fltrace)\IncludeInRelease{0000/00/00}%
1689 <(latexrelease | fltrace)  {\@addtoblcol}{float order in 2-column}%
1690 <(latexrelease | fltrace)\def\@addtoblcol{%
1691 <(latexrelease | fltrace)  \begingroup
1692   <*trace>
1693   <(latexrelease | fltrace)  \f@trace{***Start addtoblcol}%
1694   </trace>
1695   <(latexrelease | fltrace)  \cinsertfalse

```

```

1696 〈latexrelease | fltrace〉      \csetfloattypecounts
1697 〈latexrelease | fltrace〉      \cgetfpsbit \tw@
1698 〈*trace〉
1699 〈latexrelease | fltrace〉      \fl@trace{fpstype \ifodd \ctempcnta OK
1700 〈latexrelease | fltrace〉          \else not \fi dbltop: \the \cfpstype}%
1701 〈/trace〉
1702 〈latexrelease | fltrace〉      \ifodd\ctempcnta
1703 〈latexrelease | fltrace〉          \cflsetnum \cdbltopnum
1704 〈latexrelease | fltrace〉          \ifnum \cdbltopnum>\z@%
1705 〈latexrelease | fltrace〉              \ctempswafalse
1706 〈latexrelease | fltrace〉          \ifdim \cdbltoproom>\ht\currbox
1707 〈latexrelease | fltrace〉              \ctempswatrue
1708 〈*trace〉
1709 〈latexrelease | fltrace〉      \fl@trace{Space OK: \cdbltoproom =
1710 〈latexrelease | fltrace〉          \the \cdbltoproom > \the \ht \currbox
1711 〈latexrelease | fltrace〉          (dbltoproom)}%
1712 〈/trace〉
1713 〈latexrelease | fltrace〉      \else
1714 〈*trace〉
1715 〈latexrelease | fltrace〉      \fl@trace{fpstype: \the \cfpstype (addtoblc)}%
1716 〈/trace〉
1717 〈latexrelease | fltrace〉      \ifnum \cfpstype<\sixt@%
1718 〈*trace〉
1719 〈latexrelease | fltrace〉      \fl@trace{BANG float ignoring \cdbltoproom}%
1720 〈latexrelease | fltrace〉      \fl@trace{\cspaces \cdbltoproom =
1721 〈latexrelease | fltrace〉          \the \cdbltoproom.
1722 〈latexrelease | fltrace〉          Ht float: \the \ht \currbox-BANG}%
1723 〈/trace〉

```

Need to check that there is room on the page, using the local value of \textmin to make the necessary adjustment to \cdbltoproom.

```

1724 〈latexrelease | fltrace〉      \advance \cdbltoproom \textmin
1725 〈*trace〉
1726 〈latexrelease | fltrace〉      \fl@trace{Local value of texmin: \the\textmin}%
1727 〈latexrelease | fltrace〉      \fl@trace{\cspaces space on page =
1728 〈latexrelease | fltrace〉          \the \cdbltoproom.
1729 〈latexrelease | fltrace〉          Ht float: \the \ht \currbox-BANG}%
1730 〈/trace〉
1731 〈latexrelease | fltrace〉      \ifdim \cdbltoproom>\ht\currbox
1732 〈latexrelease | fltrace〉          \ctempswatrue
1733 〈*trace〉
1734 〈latexrelease | fltrace〉      \fl@trace{Space OK BANG: space on page =
1735 〈latexrelease | fltrace〉          \the\cdbltoproom > \the\ht\currbox}%
1736 〈latexrelease | fltrace〉      \else
1737 〈latexrelease | fltrace〉      \fl@trace{fpstype: \the \cfpstype}%
1738 〈latexrelease | fltrace〉      \fl@trace{Fail---no room dbltoproom-BANG?:}%
1739 〈latexrelease | fltrace〉      \fl@trace{\cspaces space on page =
1740 〈latexrelease | fltrace〉          \the \cdbltoproom.
1741 〈latexrelease | fltrace〉          Ht float: \the \ht \currbox}%
1742 〈/trace〉
1743 〈latexrelease | fltrace〉      \fi
1744 〈latexrelease | fltrace〉      \advance \cdbltoproom -\textmin
1745 〈*trace〉
1746 〈latexrelease | fltrace〉      \else

```

```

1747 〈\latexrelease | \ftrace〉          \fl@trace{fpstype: \the \@fpstype}%
1748 〈\latexrelease | \ftrace〉          \fl@trace{Fail---no room dbltoproom-ORD?:}%
1749 〈\latexrelease | \ftrace〉          \fl@trace{\@spaces \dbltoproom =
1750 〈\latexrelease | \ftrace〉          \the \dbltoproom.
1751 〈\latexrelease | \ftrace〉          Ht float: \the \ht \currbox}%
1752 〈/trace〉                         \fi
1753 〈\latexrelease | \ftrace〉          \fi
1754 〈\latexrelease | \ftrace〉          \if@tempswa
1755 〈\latexrelease | \ftrace〉          \obitor \currtype \dbldeferlist
1756 〈\latexrelease | \ftrace〉          \fl@trace{dbldeferlist:
1757 〈*trace〉                         \dbldeferlist: (before)}%
1758 〈\latexrelease | \ftrace〉          \if@test
1759 〈\latexrelease | \ftrace〉          \fl@trace{type already on list: dbldefer}%
1760 〈/trace〉                         \else
1761 〈\latexrelease | \ftrace〉          \tempdima -\ht \currbox
1762 〈*trace〉                         \advance \tempdima
1763 〈\latexrelease | \ftrace〉          -\ifx \dbltoplist \empty
1764 〈/trace〉                         \dbltextfloatsep
1765 〈\latexrelease | \ftrace〉          \else \dblfloatsep \fi
1766 〈\latexrelease | \ftrace〉          \global \advance \dbltoproom \tempdima
1767 〈\latexrelease | \ftrace〉          \global \advance \colht \tempdima
1768 〈\latexrelease | \ftrace〉          \global \advance \dbltopnum \m@ne
1769 〈\latexrelease | \ftrace〉          \global \advance \dbltoplist \currbox
1770 〈\latexrelease | \ftrace〉          \fl@trace{dbltopnum (after) =
1771 〈\latexrelease | \ftrace〉          \the \dbltopnum}%
1772 〈\latexrelease | \ftrace〉          \fl@trace{***Success: dbltop}%
1773 〈\latexrelease | \ftrace〉          \inserttrue
1774 〈\latexrelease | \ftrace〉          \fi
1775 〈*trace〉                         \fi
1776 〈\latexrelease | \ftrace〉          \else
1777 〈\latexrelease | \ftrace〉          \fl@trace{Fail: dbltopnum = \the \dbltopnum:
1778 〈\latexrelease | \ftrace〉          fpstype \the \fpstype=ORD?}%
1779 〈/trace〉                         \ifnum \fpstype<\sixt@n
1780 〈\latexrelease | \ftrace〉          \fl@trace{ERROR: !t float not successful
1781 〈\latexrelease | \ftrace〉          (addtodbcol)}%
1782 〈\latexrelease | \ftrace〉          \fi
1783 〈*trace〉                         \fi
1784 〈\latexrelease | \ftrace〉          \else
1785 〈\latexrelease | \ftrace〉          \fl@trace{Fail: dbltopnum = \the \dbltopnum:
1786 〈\latexrelease | \ftrace〉          fpstype \the \fpstype=ORD?}%
1787 〈\latexrelease | \ftrace〉          \ifnum \fpstype<\sixt@n
1788 〈\latexrelease | \ftrace〉          \fl@trace{ERROR: !t float not successful
1789 〈\latexrelease | \ftrace〉          (addtodbcol)}%
1790 〈\latexrelease | \ftrace〉          \fi
1791 〈/trace〉                         \fi
1792 〈\latexrelease | \ftrace〉          \fi
1793 〈\latexrelease | \ftrace〉          \if@insert
1794 〈\latexrelease | \ftrace〉          \else
1795 〈\latexrelease | \ftrace〉          \fl@trace{put on dbldeferlist}%
1796 〈*trace〉                         \cons\@dbldeferlist\currbox
1797 〈\latexrelease | \ftrace〉          \if@test
1798 〈/trace〉                         \cons\@dbldeferlist\currbox
1799 〈\latexrelease | \ftrace〉          \else
1800 〈*trace〉
```

```

1801 <{latexrelease | fltrace}      \f1@trace{dbldeferlist: \@dbldeferlist: (after)}%
1802 </trace>
1803 <{latexrelease | fltrace}      \fi
1804 <{*trace}>
1805 <{latexrelease | fltrace}      \f1@trace{End of addtodblcol -- locally count:}%
1806 <{latexrelease | fltrace}      \f1@trace{ dbltop: \the \@dbltopnum.}%
1807 </trace>
1808 <{latexrelease | fltrace}      \endgroup
1809 <{*trace}>
1810 <{latexrelease | fltrace}      \f1@trace{End of addtodblcol -- globally count:}%
1811 <{latexrelease | fltrace}      \f1@trace{dbltop: \the \@dbltopnum.}%
1812 </trace>
1813 <{latexrelease | fltrace}>%
1814 <{latexrelease | fltrace}\EndIncludeInRelease

```

(End of definition for \@addtodbcol.)

\@addmarginpar

```

1815 <{*2ekernel}>
1816 \def\@addmarginpar{\@next\@marbox\@currlist{\@cons\@freelist\@marbox
1817   \@cons\@freelist\@currbox}\@latexbug\@tempcnta\@ne
1818   \if@twocolumn
1819     \if@firstcolumn \@tempcnta\m@ne \fi
1820   \else
1821     \if@mparswitch
1822       \ifodd\c@page \else\@tempcnta\m@ne \fi
1823     \fi
1824     \if@reversemargin \@tempcnta -\@tempcnta \fi
1825   \fi
1826   \ifnum\@tempcnta <\z@ \global\setbox\@marbox\box\@currbox \fi
1827   \global\@tempdima\@mparbottom
1828   \advance\@tempdima -\@pageht
1829   \advance\@tempdima\ht\@marbox
1830   \ifdim\@tempdima >\z@
1831     \global\warning@no@line {Marginpar on page \thepage\space moved}%
1832   \else
1833     \global\@tempdima\z@
1834   \fi
1835   \global\@mparbottom\@pageht
1836   \global\advance\@mparbottom\@tempdima
1837   \global\advance\@mparbottom\dp\@marbox
1838   \global\advance\@mparbottom\marginparpush
1839   \advance\@tempdima -\ht\@marbox

```

Putting box movement inside the ‘marbox’:

```

1840 \global\setbox \@marbox
1841   \vbox {\vskip \@tempdima
1842     \box \@marbox}%
1843 \global \ht\@marbox \z@
1844 \global \dp\@marbox \z@

```

Sticking (rather than gluing:-) the ‘marbox’ to the line above, changed vskip to kern:

```

1845 \kern -\@pagedp
1846 \nointerlineskip
1847 \hb@xt@\columnwidth

```

```

1848      {\ifnum \c@tempcnta > \z@%
1849          \hskip\columnwidth \hskip\marginparsesep%
1850      \else%
1851          \hskip -\marginparsesep \hskip -\marginparwidth%
1852      \fi%
1853      \box\c@marbox \hss}%

```

For this reason the following code can vanish:

```

\nobreak           %% No longer needed.  CAR92/12
\vskip -\c@tempdima %% No longer needed.  CAR92/12

1854 \nointerlineskip
1855 \hbox{\vrule \c@height\z@ \c@width\z@ \c@depth\c@pagedp}%

```

(End of definition for \c@addmarginpar.)

1.1.1 Kludgeins

This part of the file is part of the implementation of the following two new commands for L^AT_EX2e.

\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 T_EX do a lot of the necessary work. All we need to put into it is the amount by which the pagegoal should be changed.

```

1856 \newinsert \@kludgeins
1857 \global\dimen@\@kludgeins \maxdimen
1858 \global\count@\@kludgeins 1000

```

(End of definition for \@kludgeins.)

\enlargethispage The user command.

```

1859 \gdef \enlargethispage {%
1860     \c@ifstar
1861     {%
1862         \c
1863             \f@trace{Enlarging page height * }%
1864         \c/trace
1865         \c@enlargepage{\hbox{\kern\p@}}%
}

```

```

1866      {%
1867  <*trace>
1868      \f@trace{Enlarging page height exactly---}%
1869  </trace>
1870      \enlargepage\empty}%
1871 }

```

(End of definition for `\enlargepage` and `\enlargepage*`.)

`\enlargepage` This actually inserts the insert, after checking for extreme values of the change.

```

1872 \gdef\enlargepage#1#2{%
1873 <*trace>
1874     \f@trace{\@spaces\@spaces by #2}%
1875 </trace>
1876     \@tempskipa#2\relax
1877     \ifdim \@tempskipa>.5\maxdimen
1878         \@latex@error{Suggested\space extra\space height\space
1879             (the\@tempskipa)\space dangerously\space
1880             large}\@eha
1881     \else
1882         \ifdim \vsize<.5\maxdimen
1883     <*trace>
1884         \f@trace {Kludgeins added--pagegoal before: \the\pagegoal}%
1885     </trace>
1886         \@bsphack
1887         \insert\kludgeins{#1\vskip-\@tempskipa}%
1888         \@esphack

```

This next bit is for tracing only:

```

1889 <*trace>
1890     \ifvmode \par
1891         \f@trace {Kludgeins added--pagegoal after: \the \pagegoal}%
1892     \fi
1893 </trace>
1894     \else
1895         \@latex@error{Page\space height\space already\space
1896             too\space large}\@eha
1897     \fi
1898     \fi
1899 }

```

(End of 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@Uppercase-letter(s)-or numbers` 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.)

```

1900 </2ekernel>
1901 <*2ekernel | latexrelease>
1902 <latexrelease>\IncludeInRelease{2021/11/15}%
1903 <latexrelease>          \ShowFloat{Show float register contents}%
1904 \def>ShowFloat#1{\begingroup
1905     \let \f@trace \f@tracemessage
1906     \f@trace{***Float #1 details:}%
1907     \ifcsname bx@#1\endcsname

```

```

1908      \expandafter\fl@ShowFloat\csname bx@\#1\endcsname
1909      \else
1910          \fl@trace{Not a float!}%
1911      \fi
1912  \endgroup
1913 }
1914 \def\fl@ShowFloat#1{%
1915     \fl@traceval{\count#1}%
1916     % this here should be interpreted on day
1917     \fl@traceval{\ht#1}%
1918     \fl@traceval{\dp#1}%
1919     \fl@traceval{\wd#1}%
1920     {\tracingonline1\showboxbreadth10\showboxdepth3\showbox#1}%
1921 }
```

Here are two definitions from `fltrace` that make the above code work:

```

1921 \def \fl@traceval #1{\fl@trace{\string #1 = \the #1}}
1922 \def \fl@tracemessage #1{{\let\@elt\empty\typeout{LaTeX2e: #1}}}
1923 {/2ekernel | latexrelease}
1924 {/latexrelease}\EndIncludeInRelease

1925 {/latexrelease}\IncludeInRelease{0000/00/00}%
1926 {/latexrelease}           {\ShowFloat}{Show float register contents}%
1927 {/latexrelease}
1928 {/latexrelease}\let\ShowFloat\@undefined
1929 {/latexrelease}\let\fl@ShowFloat\@undefined
1930 {/latexrelease}\let\fl@traceval\@undefined
1931 {/latexrelease}\let\fl@tracemessage\@undefined
1932 {/latexrelease}\EndIncludeInRelease
```

(End of 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 `LaTeX2e`.

`\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.

```
\f@trace          Set-up tracing for floats independent of other tracing as it produces mega-output. Default
\f@tracefloatoff  is no tracing.

\tracefloats      1933  {*f@trace}
\def \f@tracemessage #1{\let\@elt\empty\typeout{LaTeX2e: #1}}
\def \tracefloats{\let \f@trace \f@tracemessage}
\def \tracefloatoff {\let \f@trace \gobble}
\tracefloatoff
\def \f@traceval #1{\f@trace{\string #1 = \the #1}}
\IncludeInRelease{2015/01/01}{\tracefloatvals}%
                           {trace float vals}%
\def \tracefloatvals{%
```

As `\@dblfloatplacement` sets `\f@depth` it needs to be run inside a group, otherwise the float placement will test for the wrong value.⁵³

```
1942 \begingroup
```

When the user requests `\tracefloatvals` then they should show regardless of the tracing state, so locally we make sure that it is activated.

```
1943 \tracefloats
1944 \@dblfloatplacement
1945 \@floatplacement
1946 \f@trace{***Float placement parameters:}%
1947 \f@traceval@\colnum
1948 \f@traceval@\colroom
1949 \f@traceval@\topnum
1950 \f@traceval@\toproom
1951 \f@traceval@\botnum
1952 \f@traceval@\botroom
1953 \f@traceval@\fpmin
1954 \f@trace{\string\textration = \textfraction}%
1955 \f@traceval@\dbltopnum
1956 \f@traceval@\dbltoproom
1957 \f@trace{\string\textration = \textfraction}%
1958 \f@trace{toplist: \@toplist}%
1959 \f@trace{botlist: \@botlist}%
1960 \f@trace{midlist: \@midlist}%
1961 \f@trace{deferlist: \@deferlist}%
1962 \f@trace{dbltoplist: \@dbltoplist}%
1963 %FMi \f@trace{dbldeferlist: \@dbldeferlist}%
1964 \endgroup
1965 }
1966 \EndIncludeInRelease
1967 \IncludeInRelease{0000/00/00}{\tracefloatvals}%
                           {trace float vals}%
1968 \def \tracefloatvals{%
```

⁵³This is a somewhat questionable design.

```

1970 \begingroup
1971   \tracefloats
1972   \dblfloatplacement
1973   \floatplacement
1974   \f@trace{***Float placement parameters:}%
1975   \f@traceval\@colnum
1976   \f@traceval\@colroom
1977   \f@traceval\@topnum
1978   \f@traceval\@toproom
1979   \f@traceval\@botnum
1980   \f@traceval\@botroom
1981   \f@traceval\@fpmin
1982   \f@trace{\string\textration = \textfraction}%
1983   \f@traceval\@dbltopnum
1984   \f@traceval\@dbltoproom
1985   \f@trace{\string\textration = \textfraction}%
1986   \f@trace{toplist: \@toplist}%
1987   \f@trace{botlist: \@botlist}%
1988   \f@trace{midlist: \@midlist}%
1989   \f@trace{deferlist: \@deferlist}%
1990   \f@trace{dbltoplist: \@dbltoplist}%
1991 % next line only in old releases
1992   \f@trace{dbldeferlist: \@dbldeferlist}%
1993 \endgroup
1994 }
1995 \EndIncludeInRelease

```

We need to make sure that `fltrace` comes before `flafter` to make the tracing work.

```

1996 \Qifpackageloaded{flafter}
1997 { \PackageWarningNoLine
1998   {fltrace}{Load 'fltrace' before 'flafter'\MessageBreak
1999     Attempting to recover by reloading 'flafter')}%

```

Hide the fact that `flafter` was already loaded and then request it anew.

```

2000   \expandafter\let\csname ver@flafter.sty\endcsname\relax
2001   \def\reserved@a{\relax
2002     \expandafter\let\csname\string#1+flafter+IIR\endcsname\relax}%
2003   \reserved@a\@addtocurcol
2004   \reserved@a\@addtonextcol
2005   \RequirePackage{flafter}{}}
2006 
```

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.

```

2007 (*flafter)
2008 \providecommand\f@trace[1]{}
2009 
```

(End of definition for `\f@trace` and others.)

`\suppressfloats` Float suppression commands: these set the relevant counter globally to zero. Thus they are overridden for a particular float by an `!` specifier.

```

2010 {*2ekernel}
2011 \def \suppressfloats {%
2012   \Qifnextchar [%

```

```

2013      \@flstop
2014      {\global \@colnum \z@}%
2015  }

```

Maybe this should be a loop over #1?

```

2016 \def \@flstop [#1]{%
2017   \if t#1%
2018     \global \@topnum \z@
2019   \fi
2020   \if b#1%
2021     \global \@botnum \z@
2022   \fi
2023 }

```

(End of 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.

```

2024 \newcount \@fpstype
2025 \newdimen \@reqcolroom
2026 \newdimen \@textfloatsheight
2027 
```

(End of 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}%

2028 <*2ekernel | fltrace>
2029 \def \@fpsadddefault {%
2030 <*trace>
2031   \f@trace{fps changed from: \@fps}%
2032 </trace>
2033   \edef \@fps {\@fps\csname fps@\@capttype \endcsname}%
2034   @latex@warning {%
2035     No positions in optional float specifier.\MessageBreak
2036     Default added (so using '\@fps')}%
2037 }

```

(End of definition for \@fpsadddefault.)

\@setfloattypecounts Sets counters \@fpstype and \currtype.
BANG == bit4 of \count\@currbox = 0.

```

2038 \def \@setfloattypecounts {%
2039   \currtype \count\@currbox
2040   \@fpstype \count\@currbox
2041   \divide\currtype\@xxxii \multiply\currtype\@xxxii
2042   \advance \@fpstype -\currtype
2043   (*trace)
2044   \f@trace{(mod 32) fpstype: \the \@fpstype}%
2045   \f@trace{(mult of 32) currtype: \the \@currtype}%
2046 % Tracing only: but some should be changed into real errors/warnings?
2047   \ifnum \@fpstype<\sixt@n
2048     \ifnum \@fpstype=\z@
2049       \f@trace{ERROR: no PLACEMENT, fpstype = \the \@fpstype = 0?}%
2050     \fi
2051     \ifnum \@fpstype=\ne
2052       \f@trace{WARNING: only h, fpstype = \the \@fpstype = 1?}%
2053     \fi
2054     \f@trace{BANG float}%
2055   \else
2056     \ifnum \@fpstype=\sixt@n
2057       \f@trace{ERROR: no PLACEMENT, fpstype = \the \@fpstype = 16?}%
2058     \fi
2059     \ifnum \@fpstype=17
2060       \f@trace{WARNING: only h, fpstype = \the \@fpstype = 17?}%
2061     \fi
2062     \f@trace{ORD float}%
2063   \fi
2064   (*trace)
2065 }
2066 }/2ekernel | fltrace}

```

(End of definition for \@setfloattypecounts.)

Macros for getting, testing and setting bits of the fps.

\@getfpsbit Sets \@tempcnta to required bit of \count\@currbox.

```

2067 (*2ekernel)
2068 \def \@getfpsbit {%
2069   \@boxfpsbit \@currbox
2070 }

```

(End of definition for \@getfpsbit.)

\@boxfpsbit Used above.

```

2071 \def \@boxfpsbit #1#2{%
2072   \@tempcnta \count#1%
2073   \divide \@tempcnta #2\relax
2074 }

```

(End of definition for \@boxfpsbit.)

\@testfp New definition of the float page test.

```

2075 \def \@testfp #1{%
2076   \@boxfpsbit #18\relax % Really '#1 8' for human readers!

```

```

2077     \ifodd \@tempcnta
2078     \else
2079         \@testtrue
2080     \fi
2081 }
```

(End of definition for \testfp.)

\@setfpsbit Sets required bit of \@tempcnta (to 1).

```

2082 \def \@setfpsbit #1{%
2083     \@tempcntb \@tempcnta
2084     \divide \@tempcntb #1\relax
2085     \ifodd \@tempcntb
2086     \else
2087         \advance \@tempcnta #1\relax
2088     \fi
2089 }
2090 }</2ekernel>
```

(End of 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.

```

2091 <*2ekernel | ftrace>
2092 \def \@resethfps {%
2093     \let\reserved@a\empty
2094     \ifnum \fpstype=\one
2095         \def \reserved@a {!}{%
2096             \fpstype 17
2097         \fi
2098         \ifnum \fpstype=17
2099             \global \advance \count\currbox \tw@
2100             \@latex@warning@no@line {%
2101                 '\reserved@a h' float specifier changed to '\reserved@a ht'}%
2102         <*trace>
2103             \f@trace{%
2104                 't' added to '\reserved@a h'- new Count: \the \count\currbox}%
2105     </trace>
2106     \fi
2107 }
```

(End of 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 `\@addtobdblcol`. There may be more efficient ways to handle this, but the group seems to be the simplest.

```

2108 \def \@flsetnum #1{%
2109  {*trace}
2110   \f@trace{fpstype: \the \fpstype (flsetnum \string#1)}%
2111  {/trace}
2112  \ifnum \fpstype<\sixt@n
2113  \ifnum #1=\z@
2114  {*trace}
2115    \f@trace{BANG float resetting \string#1 to 1}%
2116  {/trace}
2117  #1\@ne
2118  \fi
2119  \fi
2120  {*trace}
2121  \f@trace{\#1 (before) = \the #1}%
2122  {/trace}
2123 }

```

(End of definition for `\@flsetnum`.)

`\@flsettextmin` This ignores `\textfraction` space restriction in case BANG.

```

2124 \def \@flsettextmin {%
2125  {*trace}
2126   \f@trace{fpstype: \the \fpstype (flsettextmin)}%
2127  {/trace}
2128  \ifnum \fpstype<\sixt@n
2129  {*trace}
2130    \f@trace{BANG ignoring textmin}%
2131  {/trace}
2132  \textmin \z@
2133  \else
2134    \textmin \textfraction\colht
2135  {*trace}
2136    \f@trace{ORD textmin = \the \textmin}%
2137  {/trace}
2138  \fi
2139 }

```

(End of 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`.

```

2140 \def \@flcheckspace #1#2{%
2141   \advance \reqcolroom
2142   \ifx #2\empty \textfloatsep \else \floatsep \fi
2143  {*trace}
2144   \f@trace{colroom = \the \reqcolroom
2145           (flcheckspace \string#1 \string#2)}%
2146   \f@trace{reqcolroom = \the \reqcolroom
2147           (flcheckspace \string#1 \string#2)}%
2148  {/trace}

```

```

2149      \ifdim \@colroom>\@reqcolroom
2150          \ifdim #1>\ht\@currbox
2151              \tempswatru
2152      <*trace>
2153          \f@trace{Space OK: #1 = \the #1 > \the \ht \@currbox
2154                                         (flcheckspace \string#1 \string#2)}%
2155      </trace>
2156      \else
2157      <*trace>
2158          \f@trace{fpstype: \the \@fpstype
2159                                         (flcheckspace \string#1 \string#2)}%
2160      </trace>
2161      \ifnum \@fpstype<\sixt@n
2162      <*trace>
2163          \f@trace{BANG float ignoring #1
2164                                         (flcheckspace \string#1 \string#2):}%
2165          \f@trace{@spaces #1 = \the #1. Ht float: \the \ht \@currbox
2166                                         BANG}%
2167      </trace>
2168      \tempswatru
2169      <*trace>
2170          \else
2171              \f@trace{Fail---no room (flcheckspace \string#1 \string#2)
2172                                         (fpstype \the \@fpstype=ORD?):}%
2173              \f@trace{@spaces #1 = \the #1. Ht float: \the \ht \@currbox
2174                                         ORD?}%
2175      </trace>
2176      \fi
2177      \fi
2178      <*trace>
2179          \else
2180              \f@trace{Fail---no room at 2nd test of colroom
2181                                         (flcheckspace \string#1 \string#2)}%
2182      </trace>
2183      \fi
2184  }
2185  </2ekernel | fltrace>

```

(End of definition for \@flcheckspace.)

\@flupdates This updates everything when a float is placed.

```

2186  <*2ekernel>
2187  \def \@flupdates #1#2#3{%
2188      \global \advance #1\m@ne
2189      \global \advance \@colnum \m@ne
2190      \tempdima -\ht\@currbox
2191      \advance \tempdima
2192          -\ifx #3\empty \textfloatsep \else \floatsep \fi
2193      \global \advance #2\tempdima
2194      \global \advance \@colroom \tempdima
2195      \cons #3\@currbox
2196  }
2197  </2ekernel>

```

(End of definition for \oflupdates.)

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, marginals, 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@parbottom \z@
  \if@twocolumn
```

```

    \@outputdblcol
\else
    \@outputpage
% This is not needed since it is done at the end of
% |\@outputpage|:
\global \@colht \textheight
\fi}

Only tracing has been added to these.

2198 <{latexrelease | fltrace}>\IncludeInRelease{2017/01/01}%
2199 <{latexrelease | fltrace}> {\@makefcolumn}{negative height floats}%
2200 <{/2ekernel | fltrace | latexrelease}%
2201 \def\@makefcolumn #1{%
2202     \begingroup
2203         \@fpmin -\maxdimen
2204         \let \@testfp \gobble
2205         \atryfcolumn #1%
2206     \endgroup
2207     {*trace}
2208     \if@fcolmade
2209         \fl@trace{PAGE: in \string\clearpage
2210                         \if@twocolumn ---twocolumn\fi---}%
2211         \fl@trace{----- float column/page completed from \string#1}%
2212     \fi
2213     /{trace}
2214 }

2215 <{latexrelease | fltrace}>\EndIncludeInRelease
2216 <{latexrelease | fltrace}>\IncludeInRelease{0000/00/00}%
2217 <{latexrelease | fltrace}> {\@makefcolumn}{negative height floats}%
2218 <{latexrelease | fltrace}>\def\@makefcolumn #1{%
2219     \begingroup
2220     \@fpmin \z@
2221     \let \@testfp \gobble
2222     \atryfcolumn #1%
2223     \endgroup
2224     {*trace}
2225     \if@fcolmade
2226         \fl@trace{PAGE: in \string\clearpage
2227                         \if@twocolumn ---twocolumn\fi---}%
2228         \fl@trace{----- float column/page completed
2229                         from \string#1}%
2230     \fi
2231     /{trace}
2232     \endgroup
2233     \EndIncludeInRelease
2234     <{/2ekernel | fltrace | latexrelease}%

```

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 `\@textbottom` adding nothing.

Thus again it is essential for `\@textbottom` to have depth 0pt.

```
2235 <{latexrelease | fltrace}>\IncludeInRelease{2015/01/01}%
```

```

2236 <{latexrelease | fltrace} {\\outputdblcol}{2 column marks}%
2237 <{*2ekernel | fltrace | latexrelease}

```

This is just a change to the single command `\@outputdblcol` so that it saves mark information for the first column and restores it in the second column.

```

2238 \\def\\@outputdblcol{%
2239   \\if@firstcolumn
2240     \\global\\@firstcolumnfalse

```

Save the left column

```

2241   \\global\\setbox\\@leftcolumn\\copy\\outputbox
2242 <{fltrace} \\f@trace{PAGE: first column boxed}%

```

Remember the marks from the first column

```

2243   \\splitmaxdepth\\maxdimen
2244   \\vbadness\\maxdimen

```

In case of `\enlargeithispage` 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 `\vspli` 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.

```

2245   \\setbox\\@outputbox\\vbox{\\unvbox\\@outputbox\\unskip}%
2246   \\setbox\\@outputbox\\vspli\\@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`.

```

2247   \\toks@\\expandafter{\\topmark}%
2248   \\xdef\\@firstcoltopmark{\\the\\toks@}%
2249   \\toks@\\expandafter{\\splitfirstmark}%
2250   \\xdef\\@firstcolfirstmark{\\the\\toks@}%

```

This test does not work if truly empty marks have been inserted, but L^AT_EX 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.)

```

2251   \\ifx\\@firstcolfirstmark\\empty
2252     \\global\\let\\@setmarks\\relax
2253   \\else
2254     \\gdef\\@setmarks{%
2255       \\let\\firstmark\\@firstcolfirstmark
2256       \\let\\topmark\\@firstcoltopmark}%
2257   \\fi

```

End of change

```

2258   \\else
2259     \\global\\@firstcolumntrue
2260     \\setbox\\@outputbox\\vbox{%
2261       \\hb@xt@\\textwidth{%
2262         \\hb@xt@\\columnwidth{\\box\\@leftcolumn \\hss}}%
2263       \\hfil

```

The color of the `\vrule` should be `\normalcolor` as to not inherit the color from the column.

```

2264   {\\normalcolor\\vrule \\@width\\columnseprule}%
2265   \\hfil
2266   \\hb@xt@\\columnwidth{\\box\\@outputbox \\hss}}}%
2267 <{fltrace} \\f@trace{PAGE: second column also boxed}%
2268   \\@combinedblfloats

```

Override current first and top with those of first column if necessary

```
2269      \@setmarks
230   End of change
231
2320      \@outputpage
2321  <fltrace>    \f1@trace{PAGE: two column page completed}%
2322      \begingroup
2323          \@dblfloatplacement
2324          \@startdblcolumn
2325          \@whilesw\if@fcolmade \fi{\@outputpage
2326  <fltrace>    \f1@trace{PAGE: double float page completed}%
2327          \@startdblcolumn}%
2328      \endgroup
2329  \fi}%
2330
2331  <| latexrelease | fltrace>\EndIncludeInRelease
2332  <| latexrelease | fltrace>\IncludeInRelease{0000/00/00}%
2333  <| latexrelease | fltrace>  {\@outputdblcol}{2 column marks}%
2334  <| latexrelease | fltrace>\def\@outputdblcol{%
2335  <| latexrelease | fltrace>  \if@firstcolumn
2336  <| latexrelease | fltrace>    \global \@firstcolumnfalse
2337  <| latexrelease | fltrace>    \global \setbox\@leftcolumn \box\@outputbox
2338  <*trace>
2339  <| latexrelease | fltrace>    \f1@trace{PAGE: first column boxed}%
2340  </trace>
2341  <| latexrelease | fltrace>  \else
2342  <| latexrelease | fltrace>    \global \@firstcolumntrue
2343  <| latexrelease | fltrace>    \setbox\@outputbox \vbox {%
2344  <| latexrelease | fltrace>                                \hb@xt@\textwidth {%
2345  <| latexrelease | fltrace>                                \hb@xt@\columnwidth {%
2346  <| latexrelease | fltrace>                                \box\@leftcolumn \hss}%
2347  <| latexrelease | fltrace>                                \hfil
2348  <| latexrelease | fltrace>                                {\normalcolor\vrule
2349  <| latexrelease | fltrace>                                \width\columnseprule}%
2350  <| latexrelease | fltrace>                                \hfil
2351  <| latexrelease | fltrace>                                \hb@xt@\columnwidth {%
2352  <| latexrelease | fltrace>                                \box\@outputbox \hss}%
2353  <| latexrelease | fltrace>                                }%
2354  <*trace>
2355  <| latexrelease | fltrace>    \f1@trace{PAGE: second column also boxed}%
2356  </trace>
2357  <| latexrelease | fltrace>    \@combinedblfloats
2358  <| latexrelease | fltrace>    \@outputpage
2359  <*trace>
2360  <| latexrelease | fltrace>    \f1@trace{PAGE: two column page completed}%
2361  </trace>
2362  <| latexrelease | fltrace>    \begingroup
2363  <| latexrelease | fltrace>    \@dblfloatplacement
2364  <| latexrelease | fltrace>    \@startdblcolumn
```

This loop could be replaced by an `\expandafter` tail recursion in `\@startdblcolumn`.

```
2365  <| latexrelease | fltrace>    \@whilesw\if@fcolmade \fi
2366  <| latexrelease | fltrace>    {\@outputpage
```

```

2317  {*trace}
2318  <|latexrelease | fltrace>      \f1@trace{PAGE: double float page completed}%
2319  </|trace>
2320  <|latexrelease | fltrace>      \@startdblcolumn}%
2321  <|latexrelease | fltrace>      \endgroup
2322  <|latexrelease | fltrace>  \fi
2323  <|latexrelease | fltrace>}%
2324  <|latexrelease | fltrace>\EndIncludeInRelease
2325  </|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.
2326 {*2ekernel}
2327 \newcount\c@topnumber
2328 \setcounter{topnumber}{2}

(End of 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.
2329 \newcommand\topfraction{.7}

(End of 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.
2330 \newcount\c@bottomnumber
2331 \setcounter{bottomnumber}{1}

(End of 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.
2332 \newcommand\bottomfraction{.3}

(End of definition for \bottomfraction.)
- \c@totalnumber This counter holds the maximum number of floats that can appear on any text page or column.
2333 \newcount\c@totalnumber
2334 \setcounter{totalnumber}{3}

(End of 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.
²³³⁵ `\newcommand\textfraction{.2}`
(End of 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.
²³³⁶ `\newcommand\floatpagefraction{.5}`
(End of definition for \floatpagefraction.)
- \cdbltopnumber** This counter holds the maximum number of double-column floats that can appear on the top of a two-column text page.
²³³⁷ `\newcount\cdbltopnumber`
²³³⁸ `\setcounter{dbltopnumber}{2}`
(End of definition for \cdbltopnumber.)
- \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.
²³³⁹ `\newcommand\dbltopfraction{.7}`
(End of 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.
²³⁴⁰ `\newcommand\dblfloatpagefraction{.5}`
(End of definition for \dblfloatpagefraction.)

Floats on a text page

\floatsep 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.

```

2341 \newskip\floatsep
2342 \newskip\textfloatsep
2343 \newskip\intextsep
2344 \setlength\floatsep {12\p@ \oplus 2\p@ \ominus 2\p@}
2345 \setlength\textfloatsep{20\p@ \oplus 2\p@ \ominus 4\p@}
2346 \setlength\intextsep {12\p@ \oplus 2\p@ \ominus 2\p@}

```

(End of definition for \floatsep, \textfloatsep, and \intextsep.)

\dblfloatsep 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.

```
2347 \newskip\dblfloatsep  
2348 \newskip\dbltextfloatsep  
2349 \setlength\dblfloatsep {12\p@ \oplus 2\p@ \ominus 2\p@}  
2350 \setlength\dbltextfloatsep{20\p@ \oplus 2\p@ \ominus 4\p@}
```

(End of 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 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.

```
2351 \newskip\@fptop  
2352 \newskip\@fpsep  
2353 \newskip\@fpbot  
2354 \setlength\@fptop{0\p@ \oplus 1fil}  
2355 \setlength\@fpsep{8\p@ \oplus 2fil}  
2356 \setlength\@fpbot{0\p@ \oplus 1fil}
```

(End of definition for \@fptop, \@fpsep, and \@fpbot.)

\@dblfpsep Double-column ‘float pages’ in two-column mode use similar parameters.

```
2357 \newskip\@dblfpsep  
2358 \newskip\@dblfpsep  
2359 \newskip\@dblfpbot  
2360 \setlength\@dblfpsep{0\p@ \oplus 1fill}  
2361 \setlength\@dblfpbot{8\p@ \oplus 2fill}  
2362 \setlength\@dblfpbot{0\p@ \oplus 1fill}
```

(End of definition for \@dblfpsep, \@dblfpbot, 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.

```
2363 \let\topfigrule=\relax  
2364 \let\botfigrule=\relax  
2365 \let\dblfigrule=\relax  
2366 {/2ekernel}
```

(End of definition for \topfigrule, \botfigrule, and \dblfigrule.)

File Z

lthyphen.dtx

This file contains the code for loading hyphenation patterns into L^AT_EX. Most of this will end up in a file called `hyphen.ltx`. If you wish to customize your L^AT_EX 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^AT_EX 2 _{ε} .

```
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_EX'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_EX run is terminated by invoking `\@@end` (which is the L^AT_EX 2 _{ε} name for T_EX'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 aa

ltfinal.dtx

1 Final settings

This section contains the final settings for L^AT_EX. 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^AT_EX 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 of 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 of definition for \newmarks.)

Allocate 3 mark classes to be used in \markboth and \markright. Should be done earlier but for that definition of \newmarks needs moving (which it should I guess).

```
21 </2ekernel>
22 <*2ekernel | latexrelease>
23 <latexrelease>\IncludeInRelease{2022/06/01}%
24 <latexrelease>           {2e-left}{Delayed legacy marks}%
25 \NewMarkClass {2e-left}
```

```

26 \NewMarkClass {2e-right}
27 \NewMarkClass {2e-right-nonempty}
No rollback really, the marks will remain.
28 </2ekernel | latexrelease>
29 <latexrelease>\EndIncludeInRelease
30 <latexrelease>\IncludeInRelease{0000/00/00}%
31 <latexrelease> {2e-left}{Delayed legacy marks}%
32 <latexrelease>
33 <latexrelease>\EndIncludeInRelease
34 <*2ekernel>

\newXeTeXintercharclass Allocate \XeTeXintercharclass types if xetex is active. previously defined in xetex.ini.
\xe@alloc@intercharclass
\@alloc@intercharclass@top
35 </2ekernel>
36 <*2ekernel | latexrelease>
37 <latexrelease>\IncludeInRelease{2015/01/01}%
38 <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).
39 \ifx\XeTeXcharclass\@undefined
40 \else
41 \ifdim\the\XeTeXversion\XeTeXrevision\p@>0.99993\p@
42   \chardef\xe@alloc@intercharclass@top=4095
43 \else
44   \chardef\xe@alloc@intercharclass@top=255
45 \fi
46 \def\newXeTeXintercharclass{%
47   \xe@alloc\XeTeXcharclass
48   \chardef\xe@alloc@intercharclass\m@ne\xe@alloc@intercharclass@top}
49 \fi
50 </2ekernel | latexrelease>
51 <latexrelease>\EndIncludeInRelease
52 <latexrelease>\IncludeInRelease{0000/00/00}%
53 <latexrelease> {\newXeTeXintercharclass}{Extended Allocation}%
54 <latexrelease> \ifx\XeTeXcharclass\@undefined
55 <latexrelease> \else
56 <latexrelease>   \def\xe@alloc@#1#2#3#4{\global\advance#1\@ne
57 <latexrelease>   \xe@ch@ck#1#4#2%
58 <latexrelease>   \allocationnumber#1%
59 <latexrelease>   \global#3#5\allocationnumber
60 <latexrelease>   \wlog{\string#5=\string#2\the\allocationnumber}}
61 <latexrelease>   \def\xe@ch@ck#1#2#3{%
62 <latexrelease>     \ifnum#1<#2\else
63 <latexrelease>       \errmessage{No room for a new #3}%
64 <latexrelease>     \fi}
65 <latexrelease>   \def\newXeTeXintercharclass{%
66 <latexrelease>     \xe@alloc@\xe@alloc@intercharclass
67 <latexrelease>           \XeTeXcharclass\chardef\@ccilv}
68 <latexrelease> \fi
69 <latexrelease>\EndIncludeInRelease
70 <*2ekernel | latexrelease>
71 <latexrelease>\IncludeInRelease{2016/02/01}%

```

```

72 〈\latexrelease〉 {\xe@alloc@intercharclass}{Start of XeTeX class allocator}%
73  \ifx\XeTeXcharclass\@undefined
74  \else
75    \countdef\xe@alloc@intercharclass=257
76    \xe@alloc@intercharclass=\z@
77  \fi
78 〈/2ekernel | \latexrelease〉
79 〈\latexrelease〉\EndIncludeInRelease
80 〈\latexrelease〉\IncludeInRelease{2015/01/01}%
81 〈\latexrelease〉 {\xe@alloc@intercharclass}{Start of XeTeX class allocator}%
82 〈\latexrelease〉 \ifx\XeTeXcharclass\@undefined
83 〈\latexrelease〉 \else
84 〈\latexrelease〉 \xe@alloc@intercharclass=\thr@@
85 〈\latexrelease〉 \fi
86 〈\latexrelease〉\EndIncludeInRelease
87 〈\latexrelease〉\IncludeInRelease{0000/00/00}%
88 〈\latexrelease〉 {\xe@alloc@intercharclass}{Start of XeTeX class allocator}%
89 〈\latexrelease〉 \ifx\XeTeXcharclass\@undefined
90 〈\latexrelease〉 \else
91 〈\latexrelease〉 \newcount\xe@alloc@intercharclass
92 〈\latexrelease〉 \xe@alloc@intercharclass=\thr@@
93 〈\latexrelease〉 \fi
94 〈\latexrelease〉\EndIncludeInRelease
95 〈*2ekernel〉

(End of definition for \newXeTeXintercharclass, \xe@alloc@intercharclass, and
 \e@alloc@intercharclass@top.)
```

trace\string_stack\string_levels Now define the Lua function to emulate \tracingstacklevels and install it in the input_level_string callback.

```

96 〈/2ekernel〉
97 〈*2ekernel | \latexrelease〉
```

In \latexrelease mode we always remove the function from the callback, then add the correct version later.

```

98 〈\latexrelease〉\ifx\directlua\@undefined
99 〈\latexrelease〉\else
100 〈\latexrelease〉 \directlua{%
101    \if luatexbase.callbacktypes['input_level_string'] and %
102      luatexbase.in_callback('input_level_string','tracingstacklevels') then
103        luatexbase.remove_from_callback('input_level_string','tracingstacklevels')
104      end}%
105 〈\latexrelease〉\fi
106 〈\latexrelease〉\IncludeInRelease{2021/06/01}{trace_stack_levels}%
107 〈\latexrelease〉          {Lua trace_stack_levels function}%
108 〈\ifx\directlua\@undefined
109 〈\else
110 〈*2ekernel〉
111 〈\expanded{%
112    \everyjob{\the\everyjob
113    \noexpand%\directlua
114 〈/2ekernel〉
115    \directlua{%
116      local function trace_stack_levels (input_ptr)
117        local tracingstacklevels = tex.count.tracingstacklevels
```

```

118         if tex.tracingmacros > 0 or input_ptr < tracingstacklevels then
119             if tracingstacklevels > 0 then
120                 if input_ptr < tracingstacklevels then
121                     return "\string\n\string~" .. string.rep(".",input_ptr)
122                 else
123                     return "\string~\string~"
124                 end
125             else
126                 return "\string\n"
127             end
128         else
129             return ""
130         end
131     end
132     \laterelease    if luatexbase.callbacktypes['input_level_string'] then
133         luatexbase.add_to_callback('input_level_string',
134             trace_stack_levels,'tracingstacklevels')
135     \laterelease    end
136     }%
137     {*2ekernel}
138     }}%
139     /2ekernel}
140 \fi
141 \laterelease\EndIncludeInRelease
142 \laterelease

```

Then for the full rollback, just do nothing, since the function was already taken out of the rollback above.

```

143 \laterelease\IncludeInRelease{0000/00/00}{trace_stack_levels}%
144 \laterelease           {Lua trace_stack_levels function}%
145 \laterelease% Nothing here
146 \laterelease\EndIncludeInRelease
147 /2ekernel | latexrelease}
148 {*2ekernel}

```

(End of definition for trace\string_stack\string_levels.)

The default values of the picture and \fbox parameters:

```

149 \unitlength = 1pt
150 \fboxsep = 3pt
151 \fboxrule = .4pt

```

The saved value of T_EX's \maxdepth:

```

152 \Qmaxdepth      = \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}

```

153 \vsize = 1000pt
154 \@colroom = \vsize
155 \@colht = \vsize

```

Initialise \textheight \textwidth and page style, to avoid internal errors if they are not set by the class.

```

156 \textheight=.5\maxdimen
157 \textwidth=\textheight
158 \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: XeTeX 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.

```

159 \ifnum 0%
160   \ifx\Umathcode\@undefined\else 1\fi
161   \ifx\XeTeXmathcode\@undefined\else 1\fi
162   >\z@
163   \message{ Unicode character data,}
164   \input{load-unicode-data}
165 {/2ekernel}
166 \langle latexrelease \rangle \IncludeInRelease{2016/02/01}%
167 \langle latexrelease \rangle {\XeTeXintercharclasses}{XeTeX character classes}%
168 \langle latexrelease \rangle \ifx\XeTeXinterchartoks\undefined
169 \langle latexrelease \rangle \else
170 \langle latexrelease \rangle \begingroup
171 \langle latexrelease \rangle \chardef\XeTeXcharclassID = 0 %
172 \langle latexrelease \rangle \chardef\XeTeXcharclassOP = 0 %
173 \langle latexrelease \rangle \chardef\XeTeXcharclassCL = 0 %
174 \langle latexrelease \rangle \chardef\XeTeXcharclassEX = 0 %
175 \langle latexrelease \rangle \chardef\XeTeXcharclassIS = 0 %
176 \langle latexrelease \rangle \chardef\XeTeXcharclassNS = 0 %
177 \langle latexrelease \rangle \chardef\XeTeXcharclassCM = 0 %
178 \langle latexrelease \rangle \input{load-unicode-xetex-classes}
179 \langle latexrelease \rangle \endgroup
180 \langle latexrelease \rangle \global\let\xtxHanGlue\undefined
181 \langle latexrelease \rangle \global\let\xtxHanSpace\undefined
182 \langle latexrelease \rangle \global\XeTeXinterchartoks 0 1 = {}
183 \langle latexrelease \rangle \global\XeTeXinterchartoks 0 2 = {}
184 \langle latexrelease \rangle \global\XeTeXinterchartoks 0 3 = {}
185 \langle latexrelease \rangle \global\XeTeXinterchartoks 1 0 = {}
186 \langle latexrelease \rangle \global\XeTeXinterchartoks 2 0 = {}
187 \langle latexrelease \rangle \global\XeTeXinterchartoks 3 0 = {}
188 \langle latexrelease \rangle \global\XeTeXinterchartoks 1 1 = {}
189 \langle latexrelease \rangle \global\XeTeXinterchartoks 1 2 = {}
190 \langle latexrelease \rangle \global\XeTeXinterchartoks 1 3 = {}
191 \langle latexrelease \rangle \global\XeTeXinterchartoks 2 1 = {}
192 \langle latexrelease \rangle \global\XeTeXinterchartoks 2 2 = {}
193 \langle latexrelease \rangle \global\XeTeXinterchartoks 2 3 = {}
194 \langle latexrelease \rangle \global\XeTeXinterchartoks 3 1 = {}
195 \langle latexrelease \rangle \global\XeTeXinterchartoks 3 2 = {}
196 \langle latexrelease \rangle \global\XeTeXinterchartoks 3 3 = {}
197 \langle latexrelease \rangle \fi
198 \langle latexrelease \rangle \EndIncludeInRelease
199 \langle latexrelease \rangle \IncludeInRelease{0000/00/00}%

```

```

200 <latexrelease> {\XeTeXintercharclasses}{XeTeX character classes}%
201 <latexrelease> \ifx\XeTeXinterchartoks\undefined
202 <latexrelease> \else
203 <latexrelease> \input{load-unicode-xetex-classes}
204 <latexrelease> \gdef\xtxHanGlue{\hskip0pt plus 0.1em\relax}
205 <latexrelease> \gdef\xtxHanSpace{\hskip0.2em plus 0.2em minus 0.1em\relax}
206 <latexrelease> \global\XeTeXinterchartoks 0 1 = {\xtxHanSpace}
207 <latexrelease> \global\XeTeXinterchartoks 0 2 = {\xtxHanSpace}
208 <latexrelease> \global\XeTeXinterchartoks 0 3 = {\nobreak\xtxHanSpace}
209 <latexrelease> \global\XeTeXinterchartoks 1 0 = {\xtxHanSpace}
210 <latexrelease> \global\XeTeXinterchartoks 2 0 = {\nobreak\xtxHanSpace}
211 <latexrelease> \global\XeTeXinterchartoks 3 0 = {\xtxHanSpace}
212 <latexrelease> \global\XeTeXinterchartoks 1 1 = {\xtxHanGlue}
213 <latexrelease> \global\XeTeXinterchartoks 1 2 = {\xtxHanGlue}
214 <latexrelease> \global\XeTeXinterchartoks 1 3 = {\nobreak\xtxHanGlue}
215 <latexrelease> \global\XeTeXinterchartoks 2 1 = {\nobreak\xtxHanGlue}
216 <latexrelease> \global\XeTeXinterchartoks 2 2 = {\nobreak\xtxHanGlue}
217 <latexrelease> \global\XeTeXinterchartoks 2 3 = {\xtxHanGlue}
218 <latexrelease> \global\XeTeXinterchartoks 3 1 = {\xtxHanGlue}
219 <latexrelease> \global\XeTeXinterchartoks 3 2 = {\xtxHanGlue}
220 <latexrelease> \global\XeTeXinterchartoks 3 3 = {\nobreak\xtxHanGlue}
221 <latexrelease> \fi
222 <latexrelease>\EndIncludeInRelease
223 <*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.

```
224 \lccode`-='`- % default hyphen char
```

The alternative is that a “traditional” engine is in use.

```
225 \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.

```

226 \def\reserved@a#1#2{%
227   @_tempcnta#1\relax
228   @_tempcntb#2\relax
229   \reserved@b
230 }
231 \def\reserved@b{%
232   \ifnum @_tempcnta> @_tempcntb\else
233     \reserved@c @_tempcnta
234     \advance @_tempcnta \ne
235     \expandafter \reserved@b
236   \fi
237 }
```

Depending on the T_EX version, we might not be allowed to do this for non-ASCII characters.

```

238 \def\reserved@c#1{%
239   \count@=#1\advance \count@ by -"20
240   \uccode#1=\count@
241   \lccode#1=#1
242 }
```

```

243 \reserved@{\`a}{\`z}
244 \reserved@{"AO}{ "BC}
245 \reserved@{"E0}{ "FF}

```

The upper case characters need their `\uccode` and `\lccode` values set, and their `\sfcodes` set to 999.

```

246 \def\reserved@c#1{%
247   \count@=\#1\advance\count@ by "20
248   \uccode#1=\#1
249   \lccode#1=\count@
250   \sfcodes#1=999
251 }
252 \reserved@{\`A}{\`Z}
253 \reserved@{"80}{ "9C}
254 \reserved@{"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.

```

255 \uccode`^Y=\`I      % dotless i
256 \lccode`^Y=\^Y     % dotless i
257 \uccode`^Z=\`J      % dotless j, ae in OT1
258 \lccode`^Z=\^Z     % dotless j, ae in OT1
259 \lccode`^9d=\`i     % dotted I
260 \uccode`^9d=\^9d   % dotted I
261 \lccode`^9e=\^9e   % d-bar
262 \uccode`^9e=\^d0   % d-bar

```

Finally here is one that helps hyphenation in the OT1 encoding.

```

263 \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).

```

264 \lccode`-\`- =\`-    % default hyphen char
265 \lccode`127=127      % alternate hyphen char
266 \lccode`23=23        % textcompwordmark in T1

```

End of the conditional to select either Unicode or T1 encoding defaults.

```

267 \fi

```

At this stage, we can install any last-minute `expl3` set-up.

```

268 \Oexpl@finalise@setup@@
269 \def\@expl@finalise@setup@@{}}

```

This is as good a place as any to active a few XeTeX-specific settings

```

270 \ifx\XeTeXuseglyphmetrics\undefined
271 \else
272   \XeTeXuseglyphmetrics=1 %
273   \XeTeXdashbreakstate=1 %
274 \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.

```

275 \InputIfFileExists{hyphen.cfg}
276   {\typeout{=====
277     Local configuration file hyphen.cfg used^^J%
278     =====}%}
279   \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
280 }
281 {\input{hyphen.ltx}}
282 \let\@addtofilelist\@gobble

\l@nohyphenation
283 \ifx\l@nohyphenation \undefined
284   \newlanguage\l@nohyphenation
285 \fi

(End of 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.

```

286 </2ekernel>
287 <*2ekernel | latexrelease>
288 <latexrelease>\IncludeInRelease{2017/04/15}%
289 <latexrelease>      {\document@default@language}{Save language for hyphenation}%
290 \let\document@default@language\m@ne
291 </2ekernel | latexrelease>
292 <latexrelease>\EndIncludeInRelease
293 <latexrelease>\IncludeInRelease{0000/00/00}%
294 <latexrelease>      {\document@default@language}{Save language for hyphenation}%
295 %
296 <latexrelease>\let\document@default@language\undefined
297 <latexrelease>\EndIncludeInRelease
298 <*2ekernel>

```

(*End of definition for \document@default@language.*)

1.5 Font loading

Fonts loaded during the formatting process might already have changed the \font@submax from 0pt to something higher. If so, we put out a bold warning.

```

299 \ifdim \font@submax >\z@
300   \font@warning{Size substitutions with differences\MessageBreak
301     up to \font@submax\space have occurred.\MessageBreak
302     \MessageBreak
303     Please check the transcript file
304     carefully\MessageBreak
305     and redo the format generation if necessary!
306     \@gobbletwo}%
307   \errhelp{Only stopped, to give you time to
308     read the above message.}%
309 \errmessage{}

```

We reset the macro. Otherwise every user will get a warning on every job.

```

310 \def\font@submax{0pt}
311 \fi

```

For pdfTeX preload and enable automatic glyph to Unicode mapping for more reliable copy and paste support.

```

312  </2ekernel>
313  <*2ekernel | latexrelease>
314  <latexrelease>\IncludeInRelease{2021/06/01}%
315  <latexrelease>          {\pdfgentounicode}{Preload glyptounicode}%
316  \ifx \pdfgentounicode \undefined \else
317  <*2ekernel>
318  \ifnum 0=0%
319  \ifdefined\pdftexversion
320  % \pdftexversion<140 does not have \pdfgentounicode, so we only check higher values
321  \ifnum \pdftexversion=140 \ifnum\pdftexrevision<22 1\fi\fi
322  \fi
323  \relax
324  </2ekernel>
325  \input glyptounicode
326  <*2ekernel>
327  \else
328  \begingroup
329  \everyeof{\noexpand}\endlinechar-1
330  \edef\x{\endgroup
331  \everyjob{\the\everyjob\@input glyptounicode }%
332  }\x
333  \fi
334  </2ekernel>
335  \pdfgentounicode=1
336  \fi
337  </2ekernel | latexrelease>
338  <latexrelease>\EndIncludeInRelease

```

When rolling back we can't unload the glyptounicode mappings, but we can reset `\pdfgentounicode` to ensure that they aren't used.

```

339 <latexrelease>\IncludeInRelease{0000/00/00}%
340 <latexrelease>          {\pdfgentounicode}{Preload glyptounicode}%
341 <latexrelease>\ifx \pdfgentounicode \undefined \else
342 <latexrelease> \pdfgentounicode=0
343 <latexrelease>\fi
344 <latexrelease>\EndIncludeInRelease
345 <*2ekernel>

```

1.6 Input encoding

Starting with the 2018 L^AT_EX 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.

```

346 </2ekernel>
347 <*2ekernel | latexrelease>

```

Check that a classic 8-bit tex engine is being used (LaTeX or PDFLaTeX).

```
348 <latexrelease>\IncludeInRelease{2018/04/01}%
349 <latexrelease>                      {\UTFviii@invalid}{UTF-8 default}%
```

Skip this section in Unicode TeX, or if MLTeX and EncTeX are enabled.

```
350 \ifnum0%
351   \ifx\Umathcode\@undefined\else 1\fi
352   \ifx\mubyte\@undefined\else 1\fi
353   \ifx\charsubdef\@undefined\else 1\fi
354   =\z@
355 \def\saved@space@catcode{10}
356 \let\@inpenc@test\relax
357 \def\IeC{%
358   \ifx\protect\@typeset@protect
359     \expandafter\@firstofone
360   \else
361     \noexpand\IeC
362   \fi
363 }
```

Make characters active for UTF-8 input formats

```
364 \tempcnta=1
365 \loop
366   \catcode\@tempcnta=13 %
367   \advance\@tempcnta\@ne %
368 \ifnum\@tempcnta<32 %
369 \repeat %
370 \catcode0=15 % null
371 \catcode9=10 % tab
372 \catcode10=12 % ctrl J
373 \catcode12=13 % ctrl L
374 \catcode13=5 % newline
375 \tempcnta=128
376 \loop
377   \catcode\@tempcnta=13
378   \advance\@tempcnta\@ne
379 \ifnum\@tempcnta<256
380 \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^AT_EX formats.

```
381 \def\UseRawInputEncoding{%
382 \let\inputencodingname\@undefined % revert
383 \let\DeclareFontEncoding\@DeclareFontEncoding@saved % revert
384 \let\DeclareUnicodeCharacter\@undefined % revert
385 \tempcnta=1
386 \loop
387   \catcode\@tempcnta=15 %
388   \advance\@tempcnta\@ne %
389 \ifnum\@tempcnta<32 %
390 \repeat %
391 \catcode0=15 % null
392 \catcode9=10 % tab
```

```

393 \catcode10=12 % ctrl J
394 \catcode12=13 % ctrl L
395 \catcode13=5 % newline
396 \tempcnta=128
397 \loop
398   \catcode\tempcnta=12
399   \advance\tempcnta\@ne
400 \ifnum\tempcnta<256
401 \repeat
402 }

```

(End of definition for \UseRawInputEncoding.)

\DeclareFontEncoding@saved Saved version of \DeclareFontEncoding@ before utf8.def modifies it for use in \UseRawInputEncoding above.

```
403 \let\DeclareFontEncoding@saved\DeclareFontEncoding@
```

(End of definition for \DeclareFontEncoding@saved.)

```

404 \edef\inputencodingname{utf8}%
405 \input{utf8.def}
406 \let\UTFviii@undefined@err@@\UTFviii@undefined@err
407 \let\UTFviii@invalid@err@@\UTFviii@invalid@err
408 \let\UTFviii@two@octets@@\UTFviii@two@octets
409 \let\UTFviii@three@octets@@\UTFviii@three@octets
410 \let\UTFviii@four@octets@@\UTFviii@four@octets
411 {2ekernel}\def\UTFviii@undefined@err#1{\gobble#1}%
412 {2ekernel}\let\UTFviii@invalid@err\string
413 {2ekernel}\let\UTFviii@two@octets\string
414 {2ekernel}\let\UTFviii@three@octets\string
415 {2ekernel}\let\UTFviii@four@octets\string
416 {2ekernel}\everyjob\expandafter{\the\everyjob
417 {2ekernel}\let\UTFviii@undefined@err\UTFviii@undefined@err@@
418 {2ekernel}\let\UTFviii@invalid@err\UTFviii@invalid@err@@
419 {2ekernel}\let\UTFviii@two@octets\UTFviii@two@octets@@
420 {2ekernel}\let\UTFviii@three@octets\UTFviii@three@octets@@
421 {2ekernel}\let\UTFviii@four@octets\UTFviii@four@octets@@
422 {2ekernel}}
423 \let@\inpcnt@test\@undefined
424 \let\@space@catcode\@undefined

```

For formats not set up for UTF-8 default, set the C0 controls to catcode 15.

```

425 \else
426 \tempcnta=0
427 \loop
428   \catcode\tempcnta=15 %
429   \advance\tempcnta\@ne %
430 \ifnum\tempcnta<32 %
431 \repeat %
432 \catcode0=15 % null
433 \catcode9=10 % tab
434 \catcode10=12 % ctrl J
435 \catcode12=13 % ctrl L
436 \catcode13=5 % newline
437 \let\UseRawInputEncoding\relax

```

This ends the skipped code in Unicode engines:

```
438 \fi
439 </2ekernel | latexrelease>
440 <latexrelease>\EndIncludeInRelease
441 <latexrelease>\IncludeInRelease{0000/00/00}%
442 <latexrelease>          {\UTFviii@invalid}{UTF-8 default}%
443 <latexrelease>  \let\UTFviii@two@octets@combine@\undefined
444 <latexrelease>  \let\UTFviii@three@octets@combine@\undefined
445 <latexrelease>  \let\UTFviii@four@octets@combine@\undefined
446 <latexrelease>  \let\UTFviii@two@octets@string@\undefined
447 <latexrelease>  \let\UTFviii@three@octets@string@\undefined
448 <latexrelease>  \let\UTFviii@four@octets@string@\undefined
449 <latexrelease>  \let\UTFviii@two@octets@noexpand@\undefined
450 <latexrelease>  \let\UTFviii@three@octets@noexpand@\undefined
451 <latexrelease>  \let\UTFviii@four@octets@noexpand@\undefined
452 <latexrelease> \@tempcnta=0
453 <latexrelease> \loop
454 <latexrelease>  \catcode@\tempcnta=15
455 <latexrelease>  \advance@\tempcnta\@ne
456 <latexrelease> \ifnum@\tempcnta<32
457 <latexrelease> \repeat %
458 <latexrelease> \catcode9=10 % tab
459 <latexrelease> \catcode10=12 % ctrl J
460 <latexrelease> \catcode12=13 % ctrl L
461 <latexrelease> \catcode13=5 % newline
462 <latexrelease> \@tempcnta=128
463 <latexrelease> \loop
464 <latexrelease> \catcode@\tempcnta=12
465 <latexrelease> \advance@\tempcnta\@ne
466 <latexrelease> \ifnum@\tempcnta<256
467 <latexrelease> \repeat
468 <latexrelease> \let\IeC@\undefined
469 <latexrelease> \def\DeclareFontEncoding@#1#2#3{%
470 <latexrelease>  \expandafter
471 <latexrelease>  \ifx\csname T@#1\endcsname\relax
472 <latexrelease>    \def\cdp@elt{\noexpand\cdp@elt}%
473 <latexrelease>    \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
474 <latexrelease>                  {\default@family}{\default@series}%
475 <latexrelease>                  {\default@shape}}%
476 <latexrelease>    \expandafter\let\csname#1-cmd\endcsname\@changed@cmd
477 <latexrelease>  \else
478 <latexrelease>    \font@info{Redeclaring font encoding #1}%
479 <latexrelease>  \fi
480 <latexrelease>  \global\@namedef{T@#1}{#2}%
481 <latexrelease>  \global\@namedef{M@#1}{\default@M#3}%
482 <latexrelease>  \xdef\LastDeclaredEncoding{#1}%
483 <latexrelease> }
484 <latexrelease> \let\UseRawInputEncoding@\undefined
485 <latexrelease> \let\DeclareFontEncoding@saved@\undefined
486 <latexrelease> \let\inputencodingname@\undefined
487 <latexrelease> \EndIncludeInRelease
```

```

488  {*2ekernel}
489 %     \begin{macrocode}
490 %
491 % We temporarily define |\reserved@a| to apply |\reserved@c| to all the
492 % numbers in the range of its arguments.
493 %     \begin{macrocode}
494 \def\reserved@a#1#2{%
495     \tempcnta#1\relax
496     \tempcntb#2\relax
497     \reserved@b
498 }
499 \def\reserved@b{%
500     \ifnum\tempcnta>\tempcntb\else
501         \reserved@c\tempcnta
502         \advance\tempcnta\one
503         \expandafter\reserved@b
504     \fi
505 }

```

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.

```

506 \catcode`\ =10
507 \catcode`\#=6
508 \catcode`\$=3
509 \catcode`\%=14
510 \catcode`\&=4
511 \catcode`\\=0
512 \catcode`\^=7
513 \catcode`\_=8
514 \catcode`\{=1
515 \catcode`\}=2
516 \catcode`\~=13
517 \catcode`\@=11
518 \catcode`\wedge I=10
519 \catcode`\wedge J=12
520 \catcode`\wedge L=13
521 \catcode`\wedge M=5

```

Set the ‘other’ catcodes.

```

522 \def\reserved@c#1{\catcode#1=12\relax}
523 \reserved@c{`}
524 \reserved@c{`}
525 \reserved@af{`}{{`?}}
526 \reserved@c{`[]}
527 \reserved@c{`]}
528 \reserved@c{``}
529 \reserved@c{`\|}

```

Set the ‘letter’ catcodes.

```

530 \def\reserved@c#1{\catcode#1=11\relax}
531 \reserved@af{`A}{`Z}
532 \reserved@af{`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.

```

533 \ifnum 0%
534   \ifx\Umathcode\@undefined\else 1\fi
535   \ifx\XeTeXmathcode\@undefined\else 1\fi
536   >\z@
537 \else
538   \def\reserved@c#1{%
539     \count@=#1\advance\count@ by -"20
540     \uccode#1=\count@
541     \lccode#1=#1
542   }
543   \reserved@a{\a}{\z}
544   \reserved@a{"A0}{ "BC}
545   \reserved@a{"E0}{ "FF}

```

The upper case characters need their \uccode and \lccode values set, and their \sfcodeset to 999.

```

546 \def\reserved@c#1{%
547   \count@=#1\advance\count@ by "20
548   \uccode#1=#1
549   \lccode#1=\count@
550   \sfcodeset=999
551 }
552 \reserved@a{\A}{\Z}
553 \reserved@a{"80}{ "9C}
554 \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.

```

555 \uccode`^Y='I      % dotless i
556 \lccode`^Y='^Y    % dotless i
557 \uccode`^Z='J      % dotless j, ae in OT1
558 \lccode`^Z='^Z    % dotless j, ae in OT1
559 \lccode`^9d='i     % dotted I
560 \uccode`^9d='^9d % dotted I
561 \lccode`^9e='^9e % d-bar
562 \uccode`^9e='^d0 % d-bar

```

Finally here is one that helps hyphenation in the OT1 encoding.

```

563 \lccode`^^[='^^[ % oe in OT1
564 \fi % End of reset block for 8-bit engines

```

\BCPdata A stub for use by babel, polyglossia, etc.

```

565 \ExplSyntaxOn
566 \newcommand*\BCPdata[1]{
567   \str_case:nn {#1}
568   {
569     { language } { en }

```

```

570     { region }   { US }
571     { script }   { Latn }
572     { tag }       { en-US }
573   }
574 }
575 \ExplSyntaxOff

```

(End of definition for \BCPdata.)

```

\MakeUppercase
\MakeLowercase
\MakeTitlecase
\NoCaseChange
\AddToNoCaseChangeList
  \CaseSwitch
\DeclareCaseChangeEquivalent
  \DeclareLowercaseMapping
\DeclareTitlecaseMapping
\DeclareUppercaseMapping
  \@uclclist

```

And whilst we're doing things with uc/lc tables, here are two commands to upper- and lower-case a string.

Wrappers around the L3 case changing functions. \protected to make them mostly safe as replacements for uppercase and \lowercase.

In

```
\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{\MakeUppercase Table of Contents}
      {\MakeUppercase Table of Contents}
```

In order to get round this, we redefine \MakeUppercase and \MakeLowercase to grab their argument and brace it.

Earlier versions needed to process \@uclclist in an \edef to handle legacy input encodings, but recent (2022) expl3 versions handle non-UTF8 text natively so we simply call the \text_...case:n functions.

```

576 \ExplSyntaxOn
577 \keys_define:nn { __kernel }
578   {
579     lang .str_set:N = \reserved@a ,
580     locale .str_set:N = \reserved@a
581   }
582 \cs_new_protected:Npn \@@text@case@aux #1#2#3
583   {
584     \cs_set_nopar:Npn \reserved@a { }
585     \tl_if_blank:nTF {#2}
586     {
587       \str_set:Nx \reserved@a
588         { \BCPdata { casing } }
589       \str_if_empty:NT \reserved@a
590         {
591           \str_set:Nx \reserved@a
592             { \BCPdata { language } }
593         }
594     }
595     { \keys_set:nn { __kernel } {#2} }
596     \use:c { text_ #1 case:Vn } \reserved@a {#3}
597   }

```

The odd use of *three* spaces here is needed as \tcmd uses the name with one and two spaces to give a ‘friendly’ error message for a runaway argument: that means we can’t use it here.

```

598 \exp_args_generate:n { cnx }
599 \cs_set_protected:Npn \reserved@a #1
600 {
601     \cs_generate_variant:cn { text_ \str_lowercase:n {#1} case:nn } { V }
602     \ExpandArgs { cnx } \NewExpandableDocumentCommand
603         { Make#1case }
604         { O{} +m }
605         { \exp_not:c { Make#1case \c_space_tl \c_space_tl \c_space_tl } [####1] {####2} }
606     }
607 \reserved@a { Upper }
608 \reserved@a { Lower }
609 \reserved@a { Title }

```

Currently, babel uses the equivalence of \oe and \OE to force casing of some material, most notably in \today. To enable that to work, we have to set those commands equal even though the current case changing code does not work using this approach.

```

610 \cs_new_protected:cpn { MakeLowercase \c_space_tl \c_space_tl \c_space_tl } [#1] #2
611 {
612     \let \OE \oe
613     \@@text@case@aux { lower } {#1} {#2}
614 }
615 \cs_new_protected:cpn { MakeUppercase \c_space_tl \c_space_tl \c_space_tl } [#1] #2
616 {
617     \let \oe \OE
618     \@@text@case@aux { upper } {#1} {#2}
619 }
620 \cs_new_protected:cpn { MakeTitlecase \c_space_tl \c_space_tl \c_space_tl } [#1] #2
621 {
622     \let \oe \OE
623     \@@text@case@aux { title } {#1} {#2}
624 }

\NoCaseChange protects its argument from the case change functions.
\AddToNoCaseChangeList Allows new commands to protect their arguments, eg
AddToNoCaseChangeList{\eqref} would protect the argument of \eqref in the same
way as the argument of \ref.

625 \cs_new_protected_nopar:Npn \AddToNoCaseChangeList
626     {\tl_put_right:Nn \l_text_case_exclude_arg_tl}
627 \AddToNoCaseChangeList{ \NoCaseChange }
628 \cs_new_protected:Npn \NoCaseChange #1 {#1}
629 \cs_new_eq:NN \CaseSwitch \text_case_switch:nnnn
630 \cs_new_eq:NN \DeclareCaseChangeEquivalent
631     \text_declare_case_equivalent:Nn
632 \NewDocumentCommand \DeclareLowercaseMapping { o m m }
633 {
634     \IfNoValueTF {#1}
635         { \text_declare_lowercase_mapping:nn }
636         { \text_declare_lowercase_mapping:nnn {#1} }
637         {#2} {#3}
638 }
639 \NewDocumentCommand \DeclareTitlecaseMapping { o m m }

```

```

640  {
641    \IfNoValueTF {#1}
642    { \text_declare_titlecase_mapping:nn }
643    { \text_declare_titlecase_mapping:nnn {#1} }
644    {#2} {#3}
645  }
646 \NewDocumentCommand \DeclareUppercaseMapping { o m m }
647  {
648    \IfNoValueTF {#1}
649    { \text_declare_uppercase_mapping:nn }
650    { \text_declare_uppercase_mapping:nnn {#1} }
651    {#2} {#3}
652  }
653 \ExplSyntaxOff
654 \def\@uclclist{\oe\OE\o\O\ae\AE
655   \dh\DH\dj\DJ\l\L\ng\NG\ss\SS\ij\IJ\th\TH}

```

(End of definition for `\MakeUppercase` and others.)

1.8 Applying Patch files

Between major releases, small patches will be distributed in files `ltpatch.ltx` which must be added at this point.

Patch file code removed.

```

656 \% \IfFileExists{ltpatch.ltx}
657 % {\typeout{=====
658 %           Applying patch file ltpatch.ltx^^J%
659 %           =====}
660 %   \def\fmtversion@topatch{unknown}
661 %   \input{ltpatch.ltx}
662 %   \ifx\fmtversion\fmtversion@topatch
663 %     \ifx\patch@level@\undefined
664 %       \typeout{^^J^^J^^J%
665 %           !!!!!!!}
666 %           !! Patch file 'ltpatch.ltx' not suitable for this^^J%
667 %           !! version of LaTeX.^^J^^J%
668 %           !! Please check if initex found an old patch file:^^J%
669 %           !! --- if so, rename it or delete it, and redo the^^J%
670 %           !! initex run.^^J%
671 %           !!!!!!!}
672 %     \batchmode \@@end
673 %   \else

```

The code below adds the ‘patch level’ string to the first `\typeout` in the startup banner.

```

674 %   \def\fmtversion@topatch{0}%
675 %   \ifx\fmtversion@topatch\patch@level\else
676 %     \def\reserved@a\typeout##1##2\reserved@a{%
677 %       \typeout{##1 patch level \patch@level}##2}
678 %       \everyjob\expandafter\expandafter\expandafter{%
679 %         \expandafter\reserved@a\the\everyjob\reserved@a}
680 %       \let\reserved@a\relax
681 %       \the\everyjob
682 %     \fi

```

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.*

```
697 \let\reserved@a@filelist  
698 \let\reserved@b=\@undefined  
699 \let\reserved@c=\@undefined  
700 \let\reserved@d=\@undefined  
701 \let\reserved@e=\@undefined  
702 \let\reserved@f=\@undefined
```

(End of definition for \reserved@a and \reserved@b.)

\toks

```
703 \toks0{  
704 \toks2{  
705 \toks4{  
706 \toks6{  
707 \toks8{
```

(End of definition for \toks.)

- `\errhelp` Empty the error help message, which may have some rubbish:

708 \errhelp{}

(End of 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`.

```
709 \def\@providesfile#1[#2]{%
710     \wlog{File: #1 #2}%
711     \expandafter\xdef\csname ver@#1\endcsname{#2}%
712 }
```

(End of definition for \@providesfile.)

\@filelist 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^AT_EX 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.

```
713 \let\@filelist\@gobble  
714 \def\@addtofilelist#1{\xdef\@filelist{\@filelist,#1}}%
```

(End of 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^AT_EX 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).

```
715 \let \IfPDFManagementActiveTF \@secondoftwo
```

(End of 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.

```
716 \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.

```
717 \makeatother  
718 \errorstopmode  
719 \dump  
720 {/2ekernel}
```

Change History

1985-11-04 ltmath.dtx LaTeX2.09		\mathversion: Test if version defined added.	470
General: produce warning message if line extends into margin. Doesn't warn about formula overprinting equation number.	703		
1989-04-10 ltfssbas.dtx v1.0a			
General: Starting with version numbers! \ifmmode added in \math@group	459		
1989-04-10 ltfssbas.dtx v1.0b			
General: \preload@sizes added.	459		
\wrong@fontshape changed to define substitution font/shape macro.	459		
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.	570		
1989-04-14 ltfssbas.dtx v1.0c			
General: More documentation added.	459		
1989-04-15 ltfssini.dtx v1.0b			
General: \mathfontset renamed to \mathversion.	570		
1989-04-19 ltfssbas.dtx v1.0d			
General: Even more doc.	459		
1989-04-21 ltfssbas.dtx v1.0e			
General: Documentation is fun! Parameters of \define@mathalphabet changed.	459		
1989-04-21 ltfssini.dtx v1.0c			
General: Changed to conform to fam.tex.	570		
1989-04-23 ltfssbas.dtx v1.0f			
General: % in \getanddefinefonts added.	459		
1989-04-26 ltfssini.dtx v1.0d			
General: \xpt added.	570		
1989-04-27 ltfssbas.dtx v1.0g			
General: Documentation revised.	459		
1989-04-27 ltfssini.dtx v1.0e			
General: Definitions of L ^A T _E X symbols corrected.	570		
1989-04-29 ltfssbas.dtx v1.0h			
General: Documented problem with \halign, and \noalign	459		
		\mathversion: Test if version defined added.	470
1989-04-29 ltfssbas.dtx v1.0i			
General: Removed the \halign \noalign correction (wasn't bugfree)	459		
1989-04-29 ltfssini.dtx v1.0f			
General: Corrections to L ^A T _E X tabular env. added.	570		
1989-05-01 ltfssbas.dtx v1.0j			
General: Default for \baselinestretch added.	459		
1989-05-22 ltfssbas.dtx v1.0k			
General: Lines longer than 72 characters folded.	459		
1989-05-22 ltfssini.dtx v1.0g			
General: Lines shortened to 72 characters	570		
1989-09-14 ltfssbas.dtx v1.0m			
General: Global replacement: \group to \mathgroup	459		
\mathversion: Corrected typo: \endcsname to \endcsmame.	470		
1989-11-07 ltfssini.dtx v1.0i			
General: All family, series, and shape names abbreviated.	570		
1989-11-08 ltfssbas.dtx v1.0o			
General: First parameter of \define@mathalphabet and \define@mathgroup changed from string to control sequence.	459		
1989-11-14 ltfssbas.dtx v1.0p			
\math@version: Math version prefix 'mv@' added.	470		
1989-11-19 ltfssbas.dtx v1.0q			
\define@newfont: Group added.	473		
\wrong@fontshape: Instead of calling \family\default@family, etc. we directly set \f@family, etc.	477		
1989-11-22 ltfssbas.dtx v1.0r			
\math@version: \def → \edef for \math@version.	470		
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.	459		
extra// → + in \extra@def.	459		

1989-11-26 ltfssbas.dtx v1.0t \select@group: \bgroup/\egroup changed to \begingroup/\endgroup to avoid empty Ord atom on math list. . .	480	1990-01-25 ltfssini.dtx v1.1e \nfss@text: Macro added.	593
1989-12-02 ltfssini.dtx v1.1b General: \rmmath renamed to \mathrm	570	1990-01-27 ltfssbas.dtx v1.2d \DeclarePreloadSizes: Font identifier set to \relax.	465
1989-12-03 ltfssini.dtx v1.1c General: Some internal macros renamed to make them inaccessible.	570	1990-01-28 ltfssbas.dtx v1.2e \mathgroup: \newfam let to \new@mathgroup.	459
1989-12-05 ltfssbas.dtx v1.0u \addto@hook: \addto@hook added. . .	485	1990-01-28 ltfssbas.dtx v1.2f \define@newfont: Added call to \curr@fontshape macro to allow substitution.	473
1989-12-05 lfsstrc.dtx v1.0u fam.dtx \every@math@size: Hook \every@size added.	517	\wrong@fontshape: Warning message slightly changed.	477
1989-12-13 lfsstrc.dtx v1.0f \use@mathgroup: \expandafter added before final \fi.	520	1990-01-28 ltfssini.dtx v1.2b \em: Call to \nomath added.	590
1989-12-16 ltfssbas.dtx v1.1a \select@group: \relax in front added.	480	1990-02-08 ltfssini.dtx v1.1g General: Protected the commands \fam, \series, \shape, \size, \selectfont, and \mathversion.	570
Now four arguments.	480	1990-02-16 ltfssbas.dtx v1.2g General: Support for changes of \baselineskip without changing the size.	459
Redefinition of alphabet now simpler.	480	\mathversion: \nomath added.	470
Usage of '=' macro added.	480	1990-02-18 lfsstrc.dtx v1.0j \selectfont: Redefine unprotected version \p@selectfont instead of \selectfont.	512
1989-12-16 lfsstrc.dtx v1.1a \selectfont: Changed order of calls.	512	1990-03-14 lfsstrc.dtx v1.0k General: Added code for TeX3.	508
\use@mathgroup: Redefinition of alphabet now simpler.	519	\extract@font: Added code for TeX3.	511
Usage of '=' macro added.	519	1990-03-30 ltfssbas.dtx v1.2h \math@egroup: Changed to have one arg.	482
1990-01-18 lfsstrc.dtx v1.0h General: \tracingfonts meaning changed.	508	1990-03-30 lfsstrc.dtx v1.2h \use@mathgroup: Third argument removed (see \math@egroup).	519
1990-01-20 ltfssbas.dtx v1.2a \math@bgroup: Def. placed in this file.	482	1990-04-01 ltfssbas.dtx v1.2i General: Code added from tracefn.tdx.	459
\math@egroup: Def. placed in this file.	482	Support for TeX3.	459
\select@group: Def for alph id changed.	480	1990-04-01 lfsstrc.dtx v1.0l General: Part of code moved to fam.dtx.	508
1990-01-21 ltfssbas.dtx v1.2b \select@group: Code moved to \use@mathgroup.	480	\tracingfonts: Check if \tracingfonts already defined.	509
1990-01-21 lfsstrc.dtx v1.2b \use@mathgroup: Macro added to allow cleaner interface.	519	1990-04-01 lfsstrc.dtx v1.0o \tracingfonts: Check if \tracingfonts defined removed again.	509
1990-01-23 ltfssbas.dtx v1.2c General: \no@version@warning renamed to \no@alphabet@error.	459		
Macro \no@alphabet@help added	459		
\no@alphabet@error: Changed to error call	459		

1990-04-02 ltfssini.dtx v1.1i General: \input of files now handled by docstrip.	570	1991-08-14 ltpictur.dtx LaTeX2.09 General: (RmS) inserted extra braces around entry for NFSS	772
1990-04-05 lfsstrc.dtx v1.0m \selectfont: Call \tracingon only if \tracingfonts greater than 3.	512	1991-08-14 ltthm.dtx LaTeX2.09 \endtheorem: Moved \itshape after \item to make it work with NFSS	802
1990-05-05 lfsstrc.dtx v1.0n \selectfont: \tracingon with new syntax.	512	1991-08-26 ltfssini.dtx v1.1n \reset@font: Macro introduced	593
1990-06-23 ltfssini.dtx v1.1k \nfss@text: Changed to \mbox.	593	1991-08-26 ltmiscen.dtx LaTeX2.09 \overline: \@par added	686
1990-06-24 ltfssbas.dtx v1.2j \DeclarePreloadSizes: Missing percent added.	465	1991-08-26 ltpictur.dtx LaTeX2.09 \endpicture: (RmS & FMi) extra boxing level around \picbox to guard against unboxing in math mode (proposed by John Hobby)	770
1990-06-24 lfsstrc.dtx v1.0o \baselinestretch: Moved to tracefnt.dtx.	517	1991-08-26 ltplain.dtx LaTeX209 \tracingall: Added \errorcontextlines=\maxdimen, suggested by J. Schrod	34
\getanddefine@fonts: \Adding tracing code.	521	1991-09-29 ltboxes.dtx LaTeX2.09 \@mpfootnotetext: (RmS) added \reset@font	737
\Macro moved from fam.dtx.	520	1991-09-29 ltfloat.dtx LaTeX2.09 \footnotetext: (RmS) added \reset@font	835
Adding debug code.	520	1991-09-29 ltmath.dtx LaTeX2.09 \@eqnnum: RmS: \reset@font added.	702
\use@mathgroup: Tracing code added.	520	1991-09-29 ltsect.dtx LaTeX2.09 \@dottedtocline: (RmS) added \reset@font for page number	814
1990-06-30 ltfssbas.dtx v1.2l \showhyphens: Macro added.	483	1991-10-17 ltcntrl.dtx LaTeX209 \@tfor: (Rms) \xdef replaced by \def (See FMi's array.doc)	327
1990-06-30 lfsstrc.dtx v1.0p \use@mathgroup: Added \relax after math group number.	520	1991-10-25 ltbibl.dtx LaTeX2.09 \citex: added \reset@font, suggested by Bernd Raichle.	844
1990-07-07 lfsstrc.dtx v1.0q \getanddefine@fonts: Group number added to tracing.	521	1991-11-01 ltfloat.dtx LaTeX2.09 \footnote: (RmS) Added \let\protect\noexpand in \footnote, \footnotemark, and \footnotetext, since \xdef is used	835
\math@egroup: Tracing code added.	520	1991-11-04 ltlists.dtx LaTeX2.09 \makelabel: (RmS) added default definition for \makelabel, to produce an error message.	722
\use@mathgroup: Group number added to tracing.	520	1991-11-04 ltplain.dtx RmS General: Removed \itemitem since never needed/useful with L ^A T _E X.	32
1990-08-27 lfsstrc.dtx 1.0r \type@restoreinfo: Some extra tracing info.	516	1991-11-06 ltbibl.dtx LaTeX2.09 \citex: added code to remove a leading blank	844
1990-08-27 lfsstrc.dtx v1.0r \getanddefine@fonts: Correcting missing name after \tracingon.	520		
1991-03-28 ltfssini.dtx v1.1m \copyright: Extra braces added.	593		
1991-03-30 ltfssini.dtx v1.2g \newfont: Definition added.	592		
\symbol: Definition added.	592		
1991-07-24 ltmiscen.dtx LaTeX2.09 \overline: (RmS) inserted extra braces around entry for NFSS	686		
1991-08-14 ltmath.dtx LaTeX2.09 \cases: (RmS) inserted extra braces around entry for NFSS	697		

1991-11-13	ltbibl.dtx	LaTeX2.09	
	\@bibitem:	Changed counter enumi to enumiv, as it says in the comment above	843
1991-11-21	ltfssini.dtx	v1.10	
	\reset@font:	Added extra braces for robustness.	593
		Changed to protected version of macro.	593
1991-11-22	ltfloat.dtx	LaTeX2.09	
	\footnote: (RmS) Added	\let\protect\noexpand in \xfootnote, \xfootnotemark, and \xfootnotetext	835
1991-11-22	ltlists.dtx	LaTeX2.09	
	\item: (RmS) Changed second call to \makelabel to \unhbox\@tempboxa. Avoids problems with side effects in \makelabel and is more efficient.	722
1991-11-27	ltfssbas.dtx	v1.3a	
	General: All \family, \shape etc. renamed to \fontfamily etc.	459	
1991-11-27	ltfssini.dtx	v1.2a	
	General: All \family, \shape etc. renamed to \fontfamily etc.	570	
1992-01-06	ltfssini.dtx	v1.2c	
	General: added slitex code	570	
1992-01-10	ltbibl.dtx	LaTeX2.09	
	\@bibitem: Changed \c@enumiv to \value of \@listctr	843	
1992-01-10	ltmath.dtx	LaTeX2.09	
	equation: RmS: put \hbox around \eqnnum to typeset the equation number in text mode (as in the eqnarray env.)	702	
1992-01-10	ltthm.dtx	LaTeX2.09	
	\@othm: (RmS) Check for existence of theorem environment	801	
1992-01-14	ltbibl.dtx	LaTeX2.09	
	\@biblabel: removed \hfill	846	
1992-01-14	ltsect.dtx	0.0	
	\@starttoc: (RmS) added \immediate to \openout as all \write commands are also executed \immediate	812	
1992-02-26	ltbibl.dtx	LaTeX2.09	
	\@lbibitem: Added \hfill to restore left-alignment of bibliography labels in alpha style	843	
1992-03-18	ltdefns.dtx	LaTeX209	
	General: (RMS) changed input channel from 0 to \inputcheck to		
	avoid conflicts with other channels allocated by \newread	82	
1992-03-18	ltfloat.dtx	LaTeX2.09	
	\@xmpar: (RmS) added \global\ignorefalse	830	
	\end@float: (RmS) changed \Eshack to \@Eshack	824	
1992-03-18	ltlists.dtx	0.0	
	\trivlist: RmS: added \nmbrlistfalse	718	
1992-03-18	ltmiscre.dtx	LaTeX2.09	
	\begin: Changed \ignoretrue to \ignorefalse (as documented)	679	
1992-03-21	ltfssini.dtx	v1.2d	
	General: Renamed \text to \nfss@text to make it internal.	570	
1992-05-12	ltfssbas.dtx	v1.3c	
	\extract@alph@from@version: Macro added.	481	
	\select@group: Added call to \extract@alph@from@version.	481	
1992-07-26	ltfssbas.dtx	v1.9a	
	\curr@fontshape:	472	
	\DeclareFontShape: Introduced \DeclareFontShape	460	
	\define@newfont:	473	
	\math@fonts:	480	
	\select@group:	480, 481	
	\split@name: Added splitting into \f@encoding.	472	
	\wrong@fontshape:	477, 478	
1992-07-26	ltfsstrc.dtx	v2.0b	
	\s@fct@:	529	
	\s@fct@sub: documentation fixes	530	
	\selectfont:	513	
	\try@simple@size:	523	
	\try@size@range:	527	
	\use@mathgroup:	520	
1992-08-14	ltbibl.dtx	LaTeX2.09	
	\@citex: added missing argument braces around \hbox, found by Ed Sznyter	844	
1992-08-14	ltboxes.dtx	LaTeX209	
	\endminipage: (RmS) replaced \vskip-\lastskip by \unskip (proposed by FMi)	737	
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)	844	
1992-08-19	ltsect.dtx	0.0	
	\@xsect: (RmS) corrected bug: stretch and shrink in argument to \hskip		

previously not negated	808	\footnote: (RmS) Changed all to ‘def’protect’noexpand’protect’noexpand	835
1992-08-19 ltthm.dtx LaTeX2.09			
\@othm: (RmS) Changed error message to complain about undefined counter	801		
1992-08-20 ltfssini.dtx v1.4b			
\@setsizE: Added \currsize.	592		
1992-08-24 ltdefns.dtx LaTeX209			
\@ifnextchar: (Rms) \@ifnextchar didn’t work if its first argument was an equal sign.	107		
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.	679		
\verb: Changed \verb and \@sverb to work correctly in math mode . . .	691		
1992-08-25 ltsect.dtx LaTeX2.09			
\@sect: (FMi) replaced explicit setting of \@svsec by call to \@seccntformat	807		
1992-09-18 ltlists.dtx LaTeX2.09			
\item: (RmS) Added warning if \item is used in math mode	720		
1992-09-18 ltab.dtx LaTeX2.09			
\@array: Changed \par to \empty to avoid starting new row e.g. after \hline	755		
1992-09-19 lfsstrc.dtx v2.0c			
\try@simple@size:	523		
1992-09-21 ltfssini.dtx v1.4d			
\not@math@alphabet: Macro defined.	591		
1992-09-22 ltfssbas.dtx v1.91a			
General: Introduced \tf@size for math size.	459		
1992-09-22 lfsstrc.dtx v2.1a			
\getanddefine@fonts: Introduced \tf@size for math size.	521		
1992-11-13 ltfssini.dtx v?			
\hexnumber@: Made expandable. . . .	593		
1992-11-23 ltcounds.dtx LaTeX209			
\stepcounter: Replaced {} in \stepcounter by \begingroup \endgroup to avoid adding an empty ord in math mode	451		
1992-11-26 ltboxes.dtx LaTeX2.09			
\@mpfootnotetext: (RmS) added protection for \edef	737		
1992-11-26 lfloat.dtx LaTeX2.09			
\@footnotetext: (RmS) added protection for \edef	835		
1992-12-03 ltfssini.dtx v?			
\hexnumber@: Make it accept counters.	593		
1993-03-08 preload.dtx v2.0b			
General: Added 12pt preloads . . .	619		
1993-03-18 ltfssbas.dtx v2.0c			
General: Changed all \tempdima in \tempdimb to avoid killing \numberline	459		
1993-03-18 lfsstrc.dtx v2.1b			
General: Changed all \tempdima in \tempdimb to avoid killing \numberline	508		
Changed all \tempdimb in \tempdimx to avoid killing \numberline	508		
1993-03-18 lfsstrc.dtx v2.1c			
\DeclareSizeFunction: Added all args to avoid blanks problems . .	526		
1993-04-09 lterror.dtx v1.0e			
\@latexerr: Mention The Companion	334		
1993-04-11 lterror.dtx v1.0f			
\@latexerr: Remove setting of errorcontextlines	334		
1993-05-05 lfntcmd.dtx v2.0b			
General: Removed all LaTeX related cmds	622		
1993-05-16 ltfssbas.dtx v2.0e			
\showhyphens: Use \reset@font . .	483		
1993-07-16 lfsstrc.dtx v2.1h			
General: Changed layout of info messages	508		
1993-07-17 ltoutenc.dtx 1.0d			
General: changed \catcoding @ . . .	406		
1993-08-03 ltmiscen.dtx LaTeX2.09			
\enddocument: Changed redefinition of \global to redefinition of \@setckpt.	672		
1993-08-05 ltpictur.dtx LaTeX2.09			
\circle: (RMS) Added error message if \circle is used in math mode. .	792		
1993-08-05 ltsect.dtx LaTeX2.09			
\@sect: (RmS) Made sure that \protect works correctly in expansion of \the counter	807		
1993-08-05 lspace.dtx LaTeX2e			
\@hspacE: (RmS) Removed superfluous \leavevmode in \hspace and \@hspacer, as suggested by CAR.	376		

1993-08-05 lttab.dtx latex2e	\everycr	772
\tabular*: Replaced \expandafter\def by \cnamedef.		754
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.	844	
1993-08-13 ltoutenc.dtx 1.0f		
General: Protected against active @ sign.	406	
1993-08-13 preload.dtx v2.0c		
General: Added \relax at end of font names.	620	
1993-08-16 ltoutenc.dtx 1.0g		
General: Needs space after \string	406	
1993-08-18 ltfssdcl.dtx v2.0e		
\new@mathversion: Exchanged names of encodings in warning message of \SetSymbolFont.	550	
1993-09-02 lfsstrc.dtx v2.1i		
General: Corrected name of sgen size function.	508	
1993-09-03 ltmiscen.dtx LaTeX2.09		
\verbatim@nolig@list: Replaced \@noligs by extensible list . . .	691	
1993-09-07 ltmiscen.dtx LaTeX2.09		
\verb@balance@group: (RmS) Changed definition of \verb so that it detects a missing second delimiter.	690	
1993-09-08 ltmiscen.dtx LaTeX2.09		
\enddocument: Added warning in case of undefined references.	672	
1993-09-15 ltfsbas.dtx v2.0g		
\DeclareFontEncoding: Corrected: \default@T to \default@M. . . .	463	
1993-09-15 lfsstrc.dtx v2.1j		
General: Corrected spelling of \noxpand.	508	
1993-09-19 lterror.dtx LaTeX2.09		
\@invalidchar: (RmS) Error message for invalid input characters. . . .	338	
1993-11-02 ltmath.dtx LaTeX2.09		
General: RmS: Corrected description of \eqnse1, moved \eqnse1 accordingly and removed extra \tabskip assignment.	703	
1993-11-03 ltmath.dtx LaTeX2e		
General: RmS: Initialized \everycr to empty	703	
1993-11-03 ltpictur.dtx LaTeX2.09		
General: (RmS) changed \halign to \ialign to initialize \tabskip and		
	\everycr	772
1993-11-11 ltfssini.dtx v2.1a	\normalfont: Macro added	593
1993-11-11 lfsstrc.dtx v2.2a		
General: Option concept added for LaTeX2e	508	
1993-11-14 ltclass.dtx v0.2a		
\currext: Name changed from \currextension	879	
\@reset@options: macro added . . .	905	
\AtEndDocument: Included extension in the generated macro name for package and class hooks.	906	
\documentstyle: Added \RequirePackage		
\@unusedoptionlist stuff. . . .	894	
\load@onefilewithoptions: Moved resetting of \default@ds, \ds@ and \@declaredoptions here, from the end of \ProcessOptions.	899	
\NeedsTeXFormat: made more robust for alternative syntax for other formats.	896	
\ProcessOptions*: Optimize ‘empty option’ code.	891	
Stop adding the global option list inside class files.	891	
1993-11-14 ltdefns.dtx v0.2a	\g@addto@macro: Made global . . .	112
1993-11-15 ltclass.dtx v0.2b		
\documentstyle: Modified to match \ProcessOption*.	894	
\ProcessOptions*: Star form added. .	891	
1993-11-17 ltclass.dtx v0.2c		
\@fileswith@ptions: Macro added	905	
\@badrequireerror: Macro added .	907	
\@twoloadclasserror: Macro added	907	
\CurrentOption: Name changed from \@curroption	879	
\DeclareOption*: Error checking added	890	
\load@onefilewithoptions: Added trap for two \LoadClass commands.	901	
\NeedsTeXFormat: Name changed from \NeedsFormat	896	
\ProcessOptions*: restoring \@fileswith@ptions added. . .	891	
1993-11-18 ltclass.dtx v0.2d		
\documentstyle: Modified \RequirePackage stuff.	894	
\ExecuteOptions: Use \CurrentOption not \reserved@a	893	

\NeedsTeXFormat: \fmtname		\newcommand: Macro reimplemented
\fmtversion not \C...	896	and extended
1993-11-21 ltfiles.dtx LaTeX2e		\renewcommand: Macro reimplemented
\@missingfileerror: Stop infinite		and extended
looping on \Cer@ext	399	\renewenvironment: Macro
1993-11-21 ltmiscen.dtx v0.9a		reimplemented and extended
\@verbatim: use \verb@font		\two@digits: Macro added
instead of \tt	687	1993-11-23 ltoutput.dtx v0.1a
\verb: Use \verb@font instead of		\paperheight: Register added
\tt.	691	\paperwidth: Register added
\verb@font: Macro added	688	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]	897	\makecol: Command changed
\@ifl@ter: Added //00 so parsing		\specialoutput: Command
never produces a runaway		changed
argument.	884	\enlargethispage*: Commands
General: \@unknownversion removed	874	added
\load@onefilewithoptions: Made the		1993-11-24 lfntcmd.dtx v2.1a
initial version [] not		\maybe@ic@: Use \t@st@ic
[\@unknownversion]	899	\t@st@ic: Macro added
1993-11-22 ltdefns.dtx LaTeX2e		1993-11-24 ltfssini.dtx v2.1a
\@minus: Macro added	81	General: Removed \xpt stuff
\@plus: Macro added	81	1993-11-24 ltlogos.dtx LaTeX2e
\CheckCommand: Macro added	88	\LaTeX: Macro changed
\providecommand: Macro added	88	1993-11-28 ltclass.dtx v0.2h
1993-11-22 lterror.dtx LaTeX2e		\@twoclasseserror: Macro added
\c@errorcontextlines: Macro added	334	General: Assorted commands now in
1993-11-22 ltfiles.dtx LaTeX2e		the kernel removed.
\listfiles: Removed checking for		Directory syntax checking moved to
\@unknownversion	401	dircheck.dtx
1993-11-22 ltlenth.dtx LaTeX2e		Primitive filenames now terminated
\@settodim: Macro added	457	by space not \relax.
\@settopoint: Macro added	458	\endfilecontents: Don't globally
\settodepth: Macro added	457	allocate a write stream (always use
\settoheight: Macro added	457	15)
1993-11-22 ltlogos.dtx LaTeX2e		1993-11-28 ltfiles.dtx LaTeX2e
\LaTeXe: Macro added	379	\@missingfileerror: Use filename
1993-11-23 ltclass.dtx v0.2g		parser from dircheck
\@use@option: Name changed from		1993-11-29 ltoutput.dtx v1.0b
\executeoption	893	\makecol: \@makespecialcolbox
General: Various macros now moved		added
to latex.tex.	878	\makespecialcolbox: Command
Warnings and errors now directly		added
coded.	878	1993-11-29 lplain.dtx LaTeX2e
1993-11-23 ltdefns.dtx LaTeX2e		General: All accents in decimals;
\@argdef: Macro added	84	suggested by Paul Taylor
\@ifundefined: Redefined to remove a		1993-11-30 ltoutput.dtx v1.0c
trailing \fi	106	\f@l@tracemessage: Commands
\@newcommand: Macro added	84	added
\@newenv: Macro interface changed	87	1993-12-01 fontdef.dtx v2.1a
\@xargdef: Macro interface changed	84	General: Update for LaTeX2e
\@yargd@f: Avoid \C?@? token	85	1993-12-01 ltoutput.dtx v1.0e
Macro interface changed	85	\reinserts: Command added

1993-12-03 ltboxes.dtx v0.1a	1993-12-05 ltfloat.dtx LaTeXe
\@argrsbox: macro removed	\@dblfloataplacement: Command
\@begin@tempboxa: macro added ..	changed 826
\@end@tempboxa: macro added ..	\@xfloat: Command changed 820
\@iirsbox: redefined to support	1993-12-05 ltoutput.dtx v1.0f
\height 740	\@addtobot: Command changed .. 1022
\@imakebox: macro modified	\@addtocurcol: Command changed 1024
\@iirsbox: redefined to support	\@addtobdblcol: Command changed 1034
\height 740	\@addtonextcol: Command changed 1031
\@isavebox: color support	\@addtotoporbot: Command
extra group 730	changed 1023
\@isavepicbox: extra group	\@boxfpsbit: Command added .. 1046
\@makebox: default changed from x to	\@flcheckspace: Command added 1048
c 727	\@flsetnum: Command added ... 1047
\@makepicbox: macro modified	\@flsettextmin: Command added 1048
\@savebox: default c not x	\@flstop: Commands added 1044
\bm@b: macros added	\@flupdates: Command added .. 1049
\endlrbox: macro added	\@fpsadddefault: Command added 1045
\fbox: extra group	\@getfpsbit: Command added .. 1046
\lrbbox: color support	\@opcol: Command changed 1007
macro added 730	Hook added 1007
\makebox: modified	\@outputpage: Command changed 1011
\mbox: extra group	\@resethfps: Command added .. 1047
\minipage: Redefined to support extra	\@setfloattypecounts: Command
optional arguments	added 1046
\newsavebox: Pass the whole of arg 1	\@setfpsbit: Command added .. 1047
to \@ifdefinable	\@shipoutsetup: Command added 1011
\parbox: Redefined to support extra	\@startcolumn: Command changed 1017
optional arguments	\@startdblcolumn: Command
\raisebox: redefined to support	changed 1017
\height 739	\@testfp: Command added
\sbox: color support	\@textfloatsheight: Commands
extra group 730	added 1045
\set@color: color support	\@topnewpage: Commands changed 999
macro added 729	\@tryfcolumn: Command changed 1018
1993-12-03 ltclass.dtx v0.2i	\@writesetup: \@startpagehook
\@cls@pkg: Name changed to avoid	added 1011
clash with output routine.	\output: Command changed 1001
General: \@onlypreamble: Many	1993-12-06 ltclass.dtx v0.2k
commands declared.	\ExecuteOptions: Preserve
Removed obsolete	\CurrentOption 893
\@documentclass	1993-12-06 ltoutput.dtx v1.0f
1993-12-03 lterror.dtx v1.0b	\@specialoutput: Unboxing of 255
\@latexerr: Set	added to rescue writes 1001
\c@errorcontextlines to -1 ..	1993-12-06 ltoutput.dtx v1.0g
1993-12-03 ltfsini.dtx v2.1a	\@topnewpage: \@floatplacement
General: update for LaTeXe	placement bug fixed 999
1993-12-04 ltfilehook.dtx v0.9b	1993-12-07 ltclass.dtx v0.2l
\unqu@tefilef@und: Macro added ..	\ProvidesFile: Macro added 888
1993-12-04 ltfiles.dtx v0.9b	1993-12-07 ltclass.dtx v0.2m
\@iinput: Macro reimplemented ..	\load@onefilewithoptions: Reset
\@input: Macro reimplemented ..	\CurrentOption 899
\IfFileExists@: Macro added	
\input: Macro reimplemented	

1993-12-07 ltoutenc.dtx 1.1		\maybe@ic@: Macro and name changed	627
General: Protected all special characters with \string.	406	\sw@slant: Macro changed	628
1993-12-07 ltoutenc.dtx v1.1		\textup: Macros changed	625
General: Made all character numbers decimal.	403	1993-12-11 ltmath.dtx v0.9g	
Removed a lot of equal signs and the like.	403	General: Added a group around the first argument of \frac to prevent changes (for example font changes) from modifying the contents of the second argument.	703
1993-12-08 ltboxes.dtx v0.1b		1993-12-11 ltoutenc.dtx v1.2a	
\@begin@tempboxa: Extra braces for color support (braces removed from other macros)	727	General: Corrected for t1enc, math.	403
\@irsbox: fix typo	740	1993-12-11 ltsect.dtx LaTeX2e	
\@parboxto: \endgraf added due to extra group in \@begin@tempboxa	734	\author: Added default	803
\lrbbox: move \endpefalse out of the inner group	730	\title: Added default	803
1993-12-08 lftntcmd.dtx v2.1b		1993-12-11 ltxref.dtx LaTeX2e	
General: Macros \rm, \bf and \sf moved to classes.dtx	630	\@setref: Macro added	666
1993-12-08 ltlists.dtx LaTeX2e		\pageref: Macro reimplemented	666
\@item: use \sbox to support colour	721	\ref: Macro reimplemented	666
1993-12-08 ltspace.dtx LaTeX2e		1993-12-12 ltoutput.dtx v1.0h	
\@bsphack: Command reimplemented	367	\@cf1b: boxmaxdepth setting moved defs changed to lets	1016
Command reimplemented; late birthday present for Chris	367	\@cf1t: name changed	1016
\@vbsphack: Command added	370	\@doclearpage: defs changed to lets	1006, 1007
1993-12-09 ltboxes.dtx v0.1c		\@startdblcolumn: defs changed to lets	1017, 1018
\@irsbox: fix another typo	740	\@topnewpage: braces removed	1000
1993-12-09 ltclass.dtx v0.2n		\@tryfcolumn: defs changed to lets	1019
\documentstyle: input 209 compatibility file.	894	\f1@tracemessage: Commands changed	1043
1993-12-09 ltfiles.dtx v0.9e		1993-12-13 ltclass.dtx v0.2o	
\document: Hook added	382	General: Removed setting \errorcontextlines (now in latex.tex)	878
1993-12-09 ltmiscen.dtx v0.9e		\documentstyle: compatibility file now latex209.sty.	894
\enddocument: Hook added	672	\usepackage: Fixed error handling	896
1993-12-10 ltoutenc.dtx v1.2		1993-12-13 ltdirchk.dtx v0.2a	
General: Added source code for t1enc.sty.	403	General: on the ‘docstrip’ pass, do not check openin path	10
1993-12-11 lftntcmd.dtx v3.0a		\IfFileExists: Removed interactive prompting for current directory syntax	10
General: Complete reworking of all text commands, using just one creator function	622	\strip@prefix: modified, name changed from \stripmeaning.	5
italic correction now put in front of penalty before glue	622	1993-12-13 ltlists.dtx latex2e	
newcommands replaced by defs	622	\trivlist: Initialised \@itemlabel	718
newfontswitch command corrected and changed	622	1993-12-13 ltmiscen.dtx v0.9h	
\DeclareTextFontCommand: Macro changed	624	\@noligs: Readded \@noligs	692
\emph: Macro changed	625	\@verbatim: Readded \@noligs	687
\fix@penalty: Macro added	628		
\maybe@ic: Macro name changed	627		

Removed optional argument of \item 686	Removed all the hackery for use in \DeclareFontEncoding, and redid everything using \DeclareTextFoo. 419, 421
center: Removed optional argument of \item 684	Removed the catcode hackery, since the file is only read as a package in the preamble, and removed all the messages on the screen, which just confuse users. Replaced them by the appropriate \ProvidesPackage commands. Added XXXenc. 406
flushleft: Removed optional argument of \item 685	
flushright: Removed optional argument of \item 686	
1993-12-13 ltoutenc.dtx v1.2b	1993-12-17 ltoutenc.dtx v1.3
General: Corrected file name in driver code. 403	General: Added \EncodingSpecificAccent, \EncodingSpecificAccentedLetter and \EncodingSpecificCommand. 403
1993-12-13 ltab.dtx latex2e	Made Rokicki's encoding a proper encoding scheme rather than a variant of OT1. 403
\tabbing: Removed optional argument of \item 749	
1993-12-14 ltoutput.dtx v1.0i	1993-12-17 ltoutput.dtx v1.0j
General: Section added to declare all parameters 1055	\copcol: Hook removed 1007
1993-12-15 ltboxes.dtx v0.1d	\@specialoutput: Page room test added 1002
\@iminipage: Changed default from 'c' to 's' 736	\@topnewpage: check for vszie too small added 999
\@iparbox: Changed default from 'c' to 's' 733	Page room test added 1001
\minipage: Changed default from 'c' to 's' 736	\@writesetup: —and then removed 1011
extra space removed. 736	\f@tracemessage: tracefloatvals made a document command ... 1043
\parbox: Changed default from 'c' to 's' 733	
1993-12-15 ltclass.dtx v0.2p	1993-12-17 ltpage.dtx LaTeX2e
General: Removed extra 's' from \@@warnings 878	\mark: Removed init \mark at begin document, since it doesn't work. 872
1993-12-16 ltlogos.dtx LaTeX2e	\rightmark: Stopgap solution to mark \leftmark and \rightmark work without initializing mark until the problem is solved. 872
\LaTeXe: Extended logo by DPC ... 379	
1993-12-16 ltmath.dtx v0.9i	1993-12-18 ltoutenc.dtx 1.3b
\@eqnacr: use \refstepcounter instead of shortcut 704	General: Fixed typos with \ProvidesPackage lines. Added the \NeedsTeXFormat line. Added the last argument to \DeclareEncoding. Moved the use of the encodings to after their declaration. 406
General: use \refstepcounter instead of shortcut 703	Replaced the missing last argument to \DeclareFontEncoding. 419, 421
1993-12-16 ltmiscen.dtx v0.9i	
General: \literal added 692	1993-12-18 ltoutenc.dtx 1.3c
1993-12-16 ltpage.dtx LaTeX2e	General: Rewrote for the new syntax of \EncodingSpecific. ... 419, 421
\mark: Init \mark at begin document 872	Split \EncodingSpecificAccent up into \EncodingSpecific and \DeclareAccent. 407
1993-12-16 ltspace.dtx LaTeX2e	
\@bsphack: Corrected optimisation :-) 367	1993-12-18 ltoutenc.dtx v1.3a
1993-12-16 ltab.dtx latex2e	General: Replaced OT3 by XXX ... 403
\@xhline: Measure from middle of vertical rules 764	
1993-12-17 ltclass.dtx v0.2q	
\@documentclasshook: Macro added 878	
\@fileswithoptions: Add \@compatibility hook 897	
\documentstyle: Match Alan's new code. 894	
1993-12-17 ltoutenc.dtx 1.3	
General: Added this section 407	

1993-12-18 ltoutenc.dtx v1.3b		1994-01-14 ltdirchk.dtx v0.2d	
General: Corrected typos.	403	\IfFileExists: Close the texsys.aux	
Replaced the missing last argument		output stream	10
to \DeclareFontEncoding.	403		
1993-12-18 ltoutenc.dtx v1.3c		1994-01-15 lfiles.dtx v0.9o	
General: A new syntax, separating		\document: move \@preamblecmds	
accent-definitions from		after document hook	384
encoding-specific definitions, and			
allowing encoding-specific			
\chardef, \let, etc.	403		
Rewrote for the new syntax of		\@fileswithoptions: Modify to	
\EncodingSpecific.	403	reduce parameter stack usage	897
1993-12-18 ltoutenc.dtx v1.3d		General: Added many more	
General: Some T1 stuff had drifted		\@onlypreamble commands	878
into the OT1 file.	403	Wrapped long lines to column 72	878
1993-12-18 lpage.dtx LaTeXe		\load@onefile@withoptions: Modify	
\sloppy: Added \emergencystretch	872	to reduce parameter stack usage	902
1993-12-19 ltclass.dtx v0.2r		1994-01-17 lfiles.dtx LaTeXe	
\endfilecontents: Different message		\listfiles: New Version, adds ‘.tex’	
when ignoring a file	907	if needed, and lines up columns	401
1993-12-19 lftncmd.dtx v3.0b		1994-01-17 lfsbas.dtx v2.1a	
General: \opdef command added . . .	622	General: New math font setup	459
Added by ASAJ.	630	\curr@math@size: New math font	
Made \newfontswitch produce an		setup	472
error if command already exists,		\everydisplay: New math font setup	471
and added \renewfontswitch,		\everymath: New math font setup	471
ASAJ	622	\frozen@everydisplay: New math	
Other tidying	622	font setup	471
Some more tidying done	622	\frozen@everymath: New math font	
Untidying added, so this is now a		setup	471
TEMPORARY version.	622	\math@version: New math font setup	470
Wording changes by CAR.	630	1994-01-17 lfssini.dtx v2.1e	
\DeclareOldFontCommand: Corrected		\not@math@alphabet: Message	
and tidied	629	changed	591
\DeclareTextFontCommand: Corrected		1994-01-17 lfsstrc.dtx v2.3a	
and tidied	624	General: New math font setup	508
1993-12-19 lspace.dtx LaTeXe		\check@mathfonts: New math font	
\bsphack: There seem to be problems		setup	518
with selfmade birthday presents .	368	\glb@currsize: New math font setup	516
1993-12-20 ltdefs.dtx LaTeXe		\restglb@settings: New math font	
\reargdef: Kept old version of		setup	519
\reargdef, for array.sty	85	1994-01-18 ltbibl.dtx LaTeXe	
1993-12-20 lfiles.dtx v0.9m		\bibliography: Use \input@ so	
\obsoletefile: Added this		include files are listed.	844
command, removed			
@oldfilewarning	401	1994-01-18 ltclass.dtx v0.2t	
1994-01-05 fontdef.dtx v2.1d		\ifclassloaded: Fix typo	
General: Removed nf prefix from file		\pkgetension	883
names.	600	1994-01-18 lfilehook.dtx v0.9p	
1994-01-13 ltmath.dtx v0.9o		\unqu@tefilef@und: New Definition	943
\eqncr: correcting 0.9i	704	1994-01-18 lfiles.dtx v0.9p	
General: correcting 0.9i	703	\iffileonpath: Macro added	397

1994-01-18 ltfssini.dtx v2.1f		1994-01-25 ltfssbas.dtx v2.1b
\@not@math@alphabet: Message corrected 591		\math@version: Corrections for math setup 470
1994-01-18 ltmiscen.dtx v0.9p		1994-01-25 ltmath.dtx LaTeXe
\@verbatim: Add \global\@inlabelfalse 686		\bordermatrix: Removed \p@renwd. 698
Only add \penalty if in hmode .. 686		
1994-01-19 fontdef.dtx v2.1e		1994-01-26 lfsstrc.dtx v2.3c
General: Added missing setting for symbols in bold version. 605		\check@mathfonts: Correct trace info placement 518
1994-01-19 ltdirchk.dtx v0.2e		\restglb@settings: Correct trace info placement 519
\IfFileExists: name changed from \test 9		1994-01-27 lfntcmd.dtx v3.1a
\input@path: No longer check that an empty group is in the path 11		\nocorrlist: Only ., used as default for cm fonts 629
\strip@prefix: name changed from \strip@meaning, to match NFSS. 5		
1994-01-19 ltmath.dtx v1.0n classes		1994-01-29 ltclass.dtx v0.2v
\mathindent: Deferred setting of \mathindent 706		\@unprocessedoptions: Macro added 907
1994-01-20 ltdirchk.dtx v0.2f		\load@onefile@withoptions: All options raise error if no \ProcessOptions appears 903
General: \copytexsys and the texsys.new file removed 9		1994-01-31 ltdefns.dtx v0.2w
Modify all of ltxcheck 14		\g@addto@macro: Use toks register to avoid ‘hash’ problems 112
\IfFileExists: \copytexsys removed 10		1994-01-31 ltfiles.dtx v0.9t
1994-01-21 ltclass.dtx v0.2u		\document: set \normalsize or \normalsize if necessary 383
\documentstyle: compatibility file now latex209.def. 894		1994-01-31 lfntcmd.dtx v3.1b
1994-01-21 ltdirchk.dtx v0.2g		General: \normalsize no longer defined 622
General: Improve documentation, reorganize docstrip module 1		1994-02-01 ltpage.dtx LaTeXe
\filename@parse: Minor changes, and add Mac version () 11		\pagestyle: (DPC) Modify to get nicer error message 869
\today: Name changed from \stamp, to save memory 9		\thispagestyle: (DPC) Modify to get nicer error message 870
1994-01-21 ltfloat.dtx LaTeXe		1994-02-02 ltclass.dtx v0.2x
\@xfloat: Added missing percent characters. 820		\load@onefile@withoptions: Only run the hook and options check if the file was loaded. 903
1994-01-21 ltmiscen.dtx v0.9s		1994-02-03 ltoutput.dtx v1.0k
\verbatim@font: Removed unnecessary category code hackery. 688		\makespecialcolbox: correct mistakes in the documentation. 1010
1994-01-24 ltdirchk.dtx v0.2h		1994-02-07 ltclass.dtx v0.2y
\IfFileExists: Stop testing once texsys.aux has been found 10		\files@withoptions: Run \compatibility on the first class to start (not the first to finish). 897
1994-01-24 ltpage.dtx LaTeXe		\@ifclasswith: Add extra ,s so ‘two’ is not matched with ‘twocolumn’ 885
\pagestyle: (DPC) Complain if pagestyle is undefined. 869		\ProcessOptions*: Add extra ,s so ‘two’ is not matched with ‘twocolumn’ 891
1994-01-25 ltdirchk.dtx v0.2i		1994-02-07 ltfssbas.dtx v2.1c
General: Protect against looping on \@input and \@end. 3		\DeclareFontEncoding: revert catcode settings earlier 463
		\DeclareFontShape@: revert catcode settings earlier 460

1994-02-08 ltoutput.dtx v1.0k		1994-03-04 ltab.dtx v1.0a	
\@makespecialcolbox: boxmaxdepth setting added	1010	General: Initial version, split from latex.dtx	742
boxmaxdepth setting removed .	1009		
General: Documentation and tasks tidied.	984		
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.	878	\@mpfootnotetext: Extra group for color	737
\@fileswithoptions: Renamed \@compatibility to \@documentclasshook. ASAJ. . .	897		
1994-02-10 ltfsbas.dtx v2.1d		1994-03-07 ltboxes.dtx v1.0a	
\addto@hook: Made \addto@hook long.	485	General: Unify format with other Kernel files	726
1994-02-10 ltfsccmp.dtx v2.1d		1994-03-07 ltdefns.dtx v1.0a	
\scan@fontshape: scan away stuff after pt	534	\@citaliccorr: Macro added	81
1994-02-22 ltfsini.dtx v2.1g		1994-03-07 ltfils.dtx v1.0a	
General: Correct error message . . .	595	General: Initial version, split from latex.dtx	380
1994-02-24 ltfsbas.dtx v2.1e		Long lines wrapped to 72 columns	380
\DeclareFontShape: Separate restoration of catcodes for fd cmds	460	1994-03-07 ltfinal.dtx v0.1a	
\define@newfont: Separate restoration of catcodes for fd cmds	473	General: Add code from the old dump.dtx	1076
\nfss@catcodes: Separate restoration of catcodes for fd cmds	474	Initial version, split from latex.dtx	1060
1994-02-25 ldirchk.dtx v0.2j		move code here from lhyphen.dtx	1066
General: Remove need for drv file . . .	1	Remove oldcomments environment	1060
1994-03-01 ldirchk.dtx v0.2k		use \InputIfFileExists not \IfFileExists	1066
General: Add unstripped module, so that dircheck.dtx may be used with initex	1	1994-03-07 ltfloat.dtx v1.0a	
1994-03-02 ltboxes.dtx v0.1e		\@endfloatbox: (DPC) Extra group for colour	825
General: Add 2ekernel module	726	\@footnotetext: (DPC) Extra group for colour	835
Remove need for drv file	726	\@xfloat: (DPC) Extra group for colour	821
1994-03-02 ltclass.dtx v0.3a		1994-03-07 lhyphen.dtx v0.1c	
General: Remove need for driver file	878	General: move the 2ekernel code to ltfinal.dtx	1058
1994-03-03 ltboxes.dtx v0.1f		1994-03-07 ltlength.dtx v1.0a	
\@irsbox: Replaced a missing \else	740	\@settodim: (DPC) Extra group for colour	457
1994-03-04 ltfloat.dtx v1.0a		1994-03-07 ltlists.dtx v1.0a	
General: Initial version, split from latex.dtx	816	General: Initial version, split from latex.dtx	710
1994-03-04 ltsect.dtx v1.0a		Long lines wrapped to 72 columns	710
General: Initial version, split from latex.dtx	803	1994-03-07 ltpage.dtx v1.0a	
		General: Initial version, split from ltherest.dtx	869
		1994-03-07 ltpictur.dtx v0.1a	
		General: Initial version, split from latex.dtx	767
		Long lines wrapped to 72 columns	767
		1994-03-07 ltsect.dtx v1.0a	
		\@hangfrom: (DPC) Extra groups for colour	810

1994-03-07 lttab.dtx v1.0a		1994-03-13 ltfilehook.dtx v0.3b	
General: Long lines wrapped to 72 columns	742	\unqu@tefilef@und: Use new cmd \@addtofilelist	943
1994-03-08 ltclass.dtx v0.3b		1994-03-13 ltfiles.dtx LaTeX2e	
General: Modify driver code into ‘new style’	878	\@addtofilelist: Macro added	401
1994-03-08 ltdirchk.dtx v1.0a		\listfiles: Reset \@addtofilelist at begin document	402
General: Reorganize driver module into ‘new style’	1	1994-03-13 ltfssbas.dtx v2.1g	
1994-03-08 ltplain.dtx v1.0a		General: add 2ekernel module to omit repeated code	459
General: Remove need for a driver file.	15	1994-03-13 ltfssdcl.dtx v2.1c	
1994-03-10 ltfssbas.dtx v2.2f		General: add 2ekernel module to omit repeated code	538
\math@egroup: Changed \begingroup/\endgroup to \bgroup/\egroup.	482	1994-03-14 ltboxes.dtx v1.0b	
1994-03-11 ltfssdcl.dtx v2.1b		\@isavebox: Use \color@setgroup	730
\DeclareSymbolFontAlphabet@: Added check against use of alphabet switch outside of math mode.	568	\@isavepicbox: Use \color@setgroup	730
\SetMathAlphabet@: Changed parameter template in temporary macro to catch check add below.	557	\color@begingroup: macro added for color support	729
1994-03-12 ltclass.dtx v0.3c		\color@endgroup: macro added for color support	729
General: Change name from docclass to ltclass	878	\lrbox: Use \color@setgroup	730
\ProvidesFile: Add \wlog	888	\sbox: Use \color@setgroup	730
\ProvidesPackage: Add \wlog	886	1994-03-14 ltfloat.dtx 1.0c	
use \gtempa	886	\@xympar: (DPC) Use \color@begingroup	830
1994-03-12 ltdefns.dtx v1.0b		1994-03-14 ltfloat.dtx v1.0c	
\@reargdef: New defn, in terms of \@yargdef	85	\@endfloatbox: (DPC) Use \color@endgroup	825
\@yargd@f: Name changed from \XXX@argdef	85	\@footnotetext: (DPC) Use \color@begingroup, add \endgraf	835
1994-03-12 ltdirchk.dtx v1.0b		\@savemarbox: (DPC) Use \color@begingroup	829
General: Change name from dircheck.dtx	1	\@xfloat: (DPC) Use \color@begingroup	821
Minor edits to the typeouts in ltxcheck	1	1994-03-15 ltfiles.dtx LaTeX2e	
1994-03-12 ltfloat.dtx v1.0b		\@missingfileerror: Quit on x or X just like a real error	399
\@savemarbox: (DPC) Extra group for colour	829	1994-03-15 lftntcmd.dtx v3.2a	
\@xympar: (DPC) Extra bgroup for colour	830	General: Adapted to mass formatting Changed \v to \@italiccorr	622
1994-03-12 ltplain.dtx v1.0b		Removed \crenewfontswitch	622
General: Name changed from lplain. The end of an era	15	Removed defs of short-forms and all sizes except \normalize	622
1994-03-12 ltplain.dtx v1.0e		1994-03-15 ltoutput.dtx v1.0l	
General: Replaced remaining width, height, depth by L ^A T _E X macro names to save tokens.	15	\@addtocr: Changed \addvspace to \vskip	1026, 1029
1994-03-13 ltcntrl.dtx v1.0c		\@combinedblfloats: Removed boxmaxdepth setting.	1017
\@tfor: (DPC) Add \@tf@r so a single group is correctly treated.	327	\@makecol: \maxdepth changed to \@maxdepth	1008
		Removed boxmaxdepth setting.	1009

\@makespecialcolbox: Removed boxmaxdepth setting.	1010	1994-03-28 ltab.dtx v1.0b General: Improve documentation	742
\@topnewpage: Corrected and amended warning message	1000	1994-03-28 ltthm.dtx v1.0a General: Initial version, split from latex.dtx	799
Warning added: it should be improved	1001	1994-03-29 ltcnts.dtx v1.0c General: Create file from parts of ltmisen and ltherest.	449
General: Added some warnings when page gets full of top floats.	984	1994-03-29 ltlenth.dtx v1.0c General: Create file ltcntlen from parts of ltmisen and ltherest.	457
Driver added and further tidying.	984	1994-03-29 ltmisen.dtx v1.0d General: Remove counter macros to ltcntlen	671
Removed duplicated code and corrected docstrip options.	984	1994-03-29 ltpageno.dtx v1.0c General: Create file ltcntlen from parts of ltmisen and ltherest.	663
Some boxmaxdepth settings removed.	984	1994-03-29 ltxref.dtx v1.0c General: Create file ltcntlen from parts of ltmisen and ltherest.	664
1994-03-16 ltclass.dtx v0.3f General: Add pkgindoc package	923	1994-03-31 ltbibl.dtx v1.0a General: Initial version of ltidxbib.dtx, split from ltherest.dtx	842
1994-03-16 ltfiles.dtx LaTeXe \listfiles: Move this code directly into \document	402	1994-03-31 ltidglo.dtx v1.0a General: Initial version of ltidxbib.dtx, split from ltherest.dtx	839
1994-03-16 ltfiles.dtx v1.0c \document: (DPC) directly add file list settings	384	1994-04-09 ltcnts.dtx v1.0d \newctr: \@nocnterr now has counter name argument	450
1994-03-16 ltmisen.dtx v1.0b \verbatim: Remove \global\@inlabelfalse again.	686	\addtocounter: \@nocnterr now has counter name argument	450
1994-03-28 ltalloc.dtx v1.0d General: Redefinition of 'new' allocations removed.	322	\setcounter: \@nocnterr now has counter name argument	450
1994-03-28 ltdirchk.dtx v1.0d General: Improve documentation	1	\stepcounter: Use \addtocounter to have name checked	451
1994-03-28 lterror.dtx v1.0d \@invalidchar: (DPC) Comment out (use catcode15 instead)	338	1994-04-09 ltthm.dtx v1.0b \othm: Use standard counter error message (FMi)	801
General: Remove test for \inputlineno undefined.	334	1994-04-11 ltclass.dtx v0.3g \endfilecontents: Add star form, don't write \endinput at the end of the file.	907
1994-03-28 ltfiles.dtx v1.0d \document: (DPC) Use \normalsize not \@normalsize	383	\ProvidesFile: Protect against weird catcodes.	888
(DPC) remove \@normalsize check	383	1994-04-11 ltfssbas.dtx v2.1h General: Added \defaultscriptratio and \defaultscriptscriptratio. ASAJ.	459
1994-03-28 ltfloat.dtx v1.0b \caption: Use \normalsize not \@normalsize	819	\defaultscriptratio: Macro added	483
General: Split further from ltherest.dtx	816	\defaultscriptscriptratio: Macro added	483
1994-03-28 ltlists.dtx v1.0b General: Improve documentation	709	1994-04-12 ltboxes.dtx v1.0c General: Remove \@acci, now defined in ltplain.dtx	734
1994-03-28 ltmisen.dtx v1.0c General: Improve Documentation	671		
1994-03-28 ltplain.dtx v1.0c \newlanguage: Remove some \outer declarations.	18		
1994-03-28 ltsect.dtx v1.0b General: Split further from ltherest.dtx	803		

Remove \@dischyp, now defined in ltinit.dtx	734	1994-04-18 ltfsstrc.dtx v2.3d General: Changed to new error/warning scheme	508
1994-04-12 ltdefns.dtx v1.0g \@dischyp: Define \@dischyp, was previously in ltboxes.dtx	110	\font@submax: Changed dimen to macro	527
1994-04-12 lplain.dtx v1.0d General: Define \@acci	33	\fontsubfuzz: Changed dimen to macro	527
1994-04-12 ltvers.dtx v1.0b General: Have version info generated automatically	38	\subst@size: \font@submax and \fontsubfuzz now macros	528
1994-04-14 lftntcmd.dtx v3.2b General: Macros renamed to non-private forms, JB	622	1994-04-19 ltpage.dtx v1.0b General: Improve documentation ..	869
\DeclareOldFontCommand: Renamed from \newfontswitch	629	1994-04-20 ltntcmd.dtx v3.3a General: Documentation up-dated ..	622
1994-04-15 ltboxes.dtx v1.0d \Qisavebox: Added missing percent character	730	New implementation of \nocorr ..	622
1994-04-17 ltcnts.dtx v1.0e \@newctr: Use \@nocounterr instead of \@nocnterr	450	\check@nocorr@: Macros added ..	626
\addtocounter: Use \@nocounterr instead of \@nocnterr	450	\maybe@ic@: \nocorr etc removed from list of tokens to check, leaving only punctuation characters	627
\setcounter: Use \@nocounterr instead of \@nocnterr	450	1994-04-20 ltmiscen.dtx v1.0e \enddocument@kernel@warnings: Changed logic for producing warning messages	674
1994-04-17 lterror.dtx v1.0h \@nocounterr: New name for error message, old error message (without arg) kept	335	1994-04-21 ltboxes.dtx v1.0e \@iiiminipage: Extra \bgroup for color	736
1994-04-17 ltthm.dtx v1.0c \@othm: Use new std counter error message (FMi)	801	\@mpfootnotetext: Extra \endgraf for color	737
1994-04-18 ltfinal.dtx v0.1b General: Initialise \textheight, \textwidth and page style ..	1063	\endminipage: Extra \egroup for color	737
1994-04-18 ltfloat.dtx v1.0d \@footnotetext: (DPC) Remove Colour support	835	1994-04-21 ltfinal.dtx v0.1c General: Added comments, set the catcodes of 128–255	1060
\@savemarbox: (DPC) Remove Colour support	829	1994-04-22 ltfsini.dtx v2.1g \not@math@alphabet: Message changed again	591
1994-04-18 ltfsbas.dtx v2.1i General: Macro \no@alphabet@help removed again	459	1994-04-23 ltfinal.dtx v0.1d General: Check that \font@submax is still zero	1060
\calculate@math@sizes: Changed message to log only	482	1994-04-24 ltoutput.dtx v1.0m \@resethfps: Number 2 changed to \tw@	1047
\no@alphabet@error: Use std LaTeX error macro	459	Warning changed	1047
1994-04-18 ltfsdcl.dtx ??? \DeclareMathAlphabet: Pass correct arg (2 not 3)	555	\@specialoutput: Message changed to give more info and ‘top’ removed	1002
1994-04-18 ltfsdcl.dtx v2.1d General: Removed surplus \no@alphabet@error (see fam.dtx)	538	\@topnewpage: Message changed to give more info	1001
		Warning message removed as it will be generated later	1000
		General: Changed \normalsize to \normalsize	984
		Corrected unverbed commands in documentation	984
		Removed some long lines and other aesthetic changes	984

Warning messages changed/corrected.	984	1994-04-30 ltfntcmd.dtx v3.3b General: Documentation up-dated and tidied	622
1994-04-24 ltpictur.dtx v0.1b General: Removed surplus spaces after \hbox to in several cases	767	Prefix frag@ changed to frag in \@protecteddef	622
1994-04-25 ltclass.dtx v0.3h General: Removed spurious extra `s at the end of error messages	878	Title changed	622
1994-04-25 ltfloat.dtx v1.0e \@largefloatcheck: Changed warning message to give more info	825	Warning changed to info message in \@protecteddef	622
Command added	825	1994-04-30 ltoutput.dtx v1.0n \@activechar@info: \@activechar@warning changed to \@activechar@info	1011
General: Changed warning messages	816	\@combinedblfloats: Removed rule in topnewpage case	1017
Removed obsolete tracing code	816	\@emptycol: Empty column action added: \@emptycol	999
1994-04-27 lfsstrc.dtx v2.3e General: Corrected item that was forgotten in last change.	508	\@fllsetnum: Rogue space removed	1048
1994-04-28 lterror.dtx v1.0j \@inmatherr: Macro added	337	\@specialoutput: Cut-off point changed to 2\baselineskip	1002
1994-04-28 lterror.dtx v1.1c \@inmatherr: Replaced \noexpand with \protect.	337	Empty column action added: \@emptycol	1002
1994-04-28 ltfssdcl.dtx v2.1e General: Removed all \uppercase in hex num parsing macros	538	Extra empty column added for twocolumn case	1002
1994-04-28 ltlists.dtx v1.0c \item: Replaced \@ltxnomath by \@inmatherr	720	Extra empty column added for twocolumn case (wrong, see below)	1002
1994-04-28 ltpictur.dtx v0.1c \@multiput: (DPC) Macro added	771	\@topnewpage: Added setting of \col@number	999, 1000
General: bezier curves added	794	Cut-off point changed to 3\baselineskip	1001
\multiput: (DPC) Ignore spaces between)(.	770	Empty column action added: \@emptycol	1001
\picture: (DPC) Ignore spaces before (.	769	Message changed for Frank	1001
1994-04-28 lplain.dtx v1.0g General: Turn off overfull box tracing in log	26	General: \@activechar@warning changed to an info message.	984
1994-04-29 ltclass.dtx v1.0a General: Change version number to 1 (no other change)	878	Added \col@number.	984
1994-04-29 ltmiscen.dtx v1.0f \@verbatim: \leavevmode added	687	Documentation tidied.	984
Change to \everypar added	687	Empty column action added.	984
1994-04-29 ltoutenc.dtx 1.4a General: Removed \EncodingSpecific. Renamed all the commands. Added \DeclareTextGlyph and \UndeclareTextCommand.	407	Fixed bug from \dblfigrule with \@topnewpage.	984
Removed Rokicki's OT1 variant encoding. Moved the driver to the top.	406	Full of floats action improved.	984
		\col@number: Added \col@number	996
		\onecolumn: Added setting of \col@number	998
		1994-05-01 lterror.dtx v1.0k \@latexerr: (CAR) Added draft \@latexinfo.	334
		1994-05-01 ltoutenc.dtx 1.4a General: Added the \a command.	415
		Added the \SaveAtCatcode and \RestoreAtCatcode commands.	419
		Removed the uc/lc table settings, since the T1 uc/lc table is now the default.	427

Rewrote for the new syntax. . .	419, 421	\end@float: (CAR) Added
1994-05-01 ltoutenc.dtx v1.4a		\@largefloatcheck 823
General: Removed Rokicki's encoding.	403	1994-05-03 ltfssdcl.dtx v2.1f
Renamed the commands, removed the \EncodingSpecific command.		General: Renamed
Turned all slots into decimal.		\@C@DeclareMathDelimiter to
Added \a.	403	\@C@DeclareMathDelimiter 538
1994-05-02 ltcntrl.dtx v1.0l		1994-05-03 ltlists.dtx v1.0d
\@break@tfor: Macro added (from ltfiles.dtx)	327	\@item: \hskip changed to \kern 721
1994-05-02 ltdefns.dtx v1.1f		\@item: Removed superfluous braces 720
\renewcommand: Removed surplus \space in error	86	1994-05-03 ltmiscen.dtx v1.0h
\renewenvironment: Removed surplus \space in error	87	\@centercr: \@badcrerr replaced by \@nolnerr 683
1994-05-02 ltfiles.dtx v1.0f		1994-05-03 ltab.dtx v1.0d
\@iffilenonpath: \@break@loop renamed to \@break@tfor	397	\@endpbox: Use \@finalstrut based on depth of \@arstrutbox 765
\@obsoletefile: Make \@onlypreamble	401	1994-05-04 ltclass.dtx v1.0b
1994-05-02 ltfinal.dtx v0.1e		\NeedsTeXFormat: Changed wording of the warning 896
General: Added setting the 'letter' catcodes.	1072	1994-05-04 lterror.dtx v1.0m
Added setting the 'other' catcodes.	1072	\@badcrerr: Error message removed 337
Added setting the special catcodes.	1072	1994-05-05 ltbibl.dtx v1.0c
Made slot 127 illegal	1072	\@citet: Set switch for warning and end of run. 844
Set all the catcodes	1060	\nocite: Do not write page number in \nocite warning message. 845
1994-05-02 ltfinal.dtx v0.1f		Set switch for warning and end of run. 845
General: Set the catcode of control-J.	1072	1994-05-05 ltfinal.dtx v0.1g
1994-05-02 ltmiscen.dtx v1.0g		General: Added empty errhelp. 1060
General: Changed 91 to 1991 and moved some bits	671	\errhelp: Set error help empty. 1077
1994-05-02 ltoutput.dtx v1.0o		1994-05-05 lfntcmd.dtx v3.3c
\@resethfps: Code shortened . . .	1047	\@C@math@egroup: Corrected \@fontswitch and added saved versions 629
General: Code of \@resethfps shortened.	984	General: Corrected \@fontswitch 622
1994-05-03 ltbibl.dtx v1.0b		1994-05-05 ltmiscen.dtx v1.0i
\nocite: Make \nocite issue a warning for an undefined citation key.	845	General: Removed braces from ifnextchar and ifstar arguments 671
1994-05-03 ltfinal.dtx v0.1f		1994-05-07 ltab.dtx v1.0c
General: Set the catcode of control-J to be 'other', for use in messages.	1060	\@maxtab: Changed \@firsttab to \chardef 746
1994-05-03 ltfloat.dtx v1.0f		Changed \@maxtab to \@chardef 746
General: (CAR) Added \@largefloatcheck	816	General: Removed definition of \+ 742
Removed unnecessary braces from arguments of \@ifnextchar	816	Removed surplus braces from \ifnextchar constructs 742
\end@dblfloat: \@largefloatcheck added	824	1994-05-08 lfntcmd.dtx v3.3d
		General: Removed \@undefinedfonterror 622
		\normalsize: Removed \@undefinedfonterror 630
		1994-05-09 lfntcmd.dtx v3.3f
		General: Replaced all \next by \let@token and undo change 3.3e, whatever that was. 622

1994-05-10 ltdefns.dtx v1.0n	\@changed@cmd and \DeclareProtectedCommand.	407
General: (ASAJ) Added \DeclareProtectedCommand.	Renamed the commands again.	
Added \DeclareProtectedCommand	Made the encoding part of the	
Removed braces around	command syntax. Added the	
\@ifundefined argument. ASAJ.	\DeclareTextCommand interface.	
\makeatother: Added \makeatletter	Used	
and \makeatother ASAJ.	\DeclareProtectedCommand.	403
1994-05-10 lterror.dtx v1.0n	\DeclareTextAccent: Reimplemented	
\@latexerr: (ASAJ) Added extra	using \DeclareTextCommand.	409
blank lines to \@latexerr.		
1994-05-10 ltmiscen.dtx v1.0j	1994-05-11 ltspace.dtx v1.0o	
\@sverb: Slight change in error	\hspace: Use	
message text.	\DeclareRobustCommand. ASAJ.	376
1994-05-11 ltboxes.dtx v1.0f	1994-05-12 ltboxes.dtx v1.0g	
\@begin@tempboxa: Use new	\@finalstrut: macro added	740
\color@setgroup concept.	\fbox: New definition, merged with	
\@iiminipage: Use new	\fframebox	731
\color@setgroup concept.	\fframebox: Merged \fbox and	
\@mpfootnotetext: Use new	\fframebox	732
\color@setgroup concept.	\normalcolor: macro added for color	
Use new \normalcolor and	support	729
\@finalstrut.		
General: Superfluous braces removed	1994-05-12 ltdefns.dtx v1.0p	
from several commands	General: (ASAJ) Fixed a bug with	
\color@setgroup: macro added for	\relax which was using \gobble	
color support	before defining it.	80
\endminipage: Use new	Fixed a bug with \relax which	
\color@setgroup concept.	was using \gobble before defining	
1994-05-11 ltclass.dtx v1.0c	it.	90
\endfilecontents: Add checks for	1994-05-12 ltfssbas.dtx v2.1j	
form feed and tab	General: New baselinestretch concept	459
1994-05-11 ltdirchk.dtx v1.0e	Replaced hand-protected commands	
General: Add \ProvidesFile as used	by \DeclareRobustCommand defs	459
in fd files.	\f@linespread: New macro	469
1994-05-11 lterror.dtx v1.0o	\fontencoding: Use	
\@latexerr: (ASAJ) Removed one of	\DeclareRobustCommand.	467
the extra blank lines to	\fontfamily: Use	
\@latexerr.	\DeclareRobustCommand.	468
1994-05-11 ltlogos.dtx v1.0o	\fontseries: Use	
\LaTeXe: Use	\DeclareRobustCommand.	468
\DeclareProtectedCommand.	\fontshape: Use	
ASAJ.	\DeclareRobustCommand.	468
\LaTeXe: Use	\fontsize: Redefined to use	
\DeclareProtectedCommand.	\set@fontsize	469
ASAJ.	\linespread: New macro	469
1994-05-11 ltoutenc.dtx 1.5a	\mathversion: Use	
General: Made T1 and OT1 generate	\DeclareRobustCommand.	470
packages rather than def files.		
Renamed the ‘package’ module to	1994-05-12 ltfssdcl.dtx v2.1g	
‘teststy’.	General: Allow \relax as undefined	
1994-05-11 ltoutenc.dtx v1.5a	command	538
General: Reimplemented	Allow \relax’ed cmds to be	
\DeclareTextCommand using	declared	538
	1994-05-12 ltfssini.dtx v2.1i	
	General: Moved \fontencoding to	
	fam.dtx	570

Moved \fontfamily to fam.dtx	570	1994-05-13 ltfiles.dtx v1.0g
Moved \fontseries to fam.dtx	570	\document: Added execution of \every@size
Moved \fontshape to fam.dtx	570	1994-05-13 ltfinal.dtx v0.1h
Moved \fontsize to fam.dtx	570	General: Added package ot1enc, and defined \@acci, \@accii and \@acciii.
Moved \selectfont to tracefnt.dtx	570	1060
1994-05-12 lfsstrc.dtx v2.3f		
\selectfont: Use \DeclareRobustCommand	512	1994-05-13 ltfinal.dtx v1.0h
1994-05-12 ltoutenc.dtx 1.5a		General: Added output enc stuff
General: Removed the \SaveAtCatcode and \RestoreAtCatcode commands.	419	1076
Rewrote for the new syntax.	419, 421	
1994-05-12 ltoutput.dtx v1.0p		1994-05-13 ltfloat.dtx v1.0g
\@writesetup: \normalcoloradded	1011	\@footnotetext: (DPC) Add new style colour support: \normalcolor
General: \normalcoloradded in various places (DPC).	984	(DPC) Use \@finalstrut
1994-05-13 ltboxes.dtx v1.0h		\@xfloat: (DPC) Use \normalcolor
\@arrayparboxrestore: New accent system, use \let not \def	735	1994-05-13 ltftcmd.dtx v3.3g
1994-05-13 ltcnts.dtx v1.0f		General: Replaced \@protecteddef by \DeclareRobustCommand
General: Removed \@Ialph	454	622
Removed \@i alph	454	1994-05-13 ltfsbas.dtx v2.1k
1994-05-13 ltdefns.dtx v1.0q		General: Remove File identification 'typeout'
General: (ASAJ) Renamed \DeclareProtectedCommand to \DeclareRobustCommand.		459
Removed \@if@short@command.	80	1994-05-13 ltfsbas.dtx v2.1l
(ASAJ) Replaces \space by ' in \csname.	80	\DeclareFontEncoding: Init encoding change command
Renamed \DeclareProtectedCommand to \DeclareRobustCommand.		463
Removed \@if@short@command.		\define@newfont: Use \@input@ for fd files
Moved to after the definition of \@gobble.	90	1994-05-13 ltfsdcl.dtx v2.1h
1994-05-13 ltdefns.dtx v1.0r		General: Removed file identification typeout
General: (ASAJ) Added logging message to \DeclareProtectedCommand.	80	538
Added logging message to \DeclareProtectedCommand.	90	1994-05-13 ltfsini.dtx v2.1j
1994-05-13 ltdefns.dtx v1.0s		General: Removed file identification typeout
General: (ASAJ) Added \@backslashchar.	80	570
(ASAJ) Coded \ifdefinable more efficiently.	80	1994-05-13 ltfsstrc.dtx v2.3g
Coded more efficiently, thanks to FMI.	86	General: Removed typeouts as \ProvidesPackage writes to log.
1994-05-13 ltfiles.dtx LaTeXe		508
\listfiles: Stop \listfiles being run twice	401	1994-05-13 ltoutenc.dtx v1.5b
		General: Added \{, \} and \\$.
		403
		Renamed \DeclareProtectedCommand to \DeclareRobustCommand.
		403
		Replaces \space by ' in \csname.
		403
		1994-05-13 ltpictur.dtx v0.1d
		General: Removed surplus braces from \@if.. constructions
		767
		1994-05-13 ltab.dtx v1.0d
		\@contfield: Colour support
		748
		\@startfield: Colour support
		748
		\@stopfield: Colour support
		748
		\@a: moved to ltoutenc
		746
		1994-05-14 fontdef.dtx v2.1f
		General: Removed .def files.
		600
		1994-05-14 ltfsbas.dtx v2.1m
		\enc@update: Macro added
		468

1994-05-14 ltfssbas.dtx v2.1n	1994-05-16 ltfssbas.dtx v2.1p	
General: Set defaults for all <code>\f@...</code>	469	
<code>\DeclareErrorFont</code> : Don't set <code>\f@encoding</code>	477	
<code>\DeclareFontEncoding</code> : Log if encoding is redeclared	463	
Only init enc change cmd when new encoding	463	
1994-05-14 ltfssini.dtx v2.1k	1994-05-16 ltlogos.dtx v1.1a	
General: Init error font just before checking for fontdef.cfg	595	
<code>\reset@font</code> : Remove surplus braces	593	
1994-05-14 ltfssrc.dtx v2.3h	1994-05-16 ltmath.dtx v1.0k	
<code>\selectfont</code> : Added <code>\enc@update</code> .	514	
1994-05-14 ltoutenc.dtx 1.5d	1994-05-16 ltoutenc.dtx 1.5h	
General: Moved the driver to the top.	406	
1994-05-14 ltoutenc.dtx v1.5c	1994-05-16 ltoutenc.dtx v1.5f	
General: Added the fontenc package	446	
Added the fontenc package.	403	
Fixed a bug which caused an infinite loop if <code>\f@encoding</code> was incorrectly set.	403, 407	
Moved fontsmpl to its own dtx file.	403	
1994-05-14 ltoutenc.dtx v1.5d	1994-05-16 ltoutenc.dtx v1.5g	
General: Rewrote <code>\DeclareTextCommand</code> to define its argument to use the current encoding by default, rather than the encoding provided to <code>\DeclareTextCommand</code>	403, 407	
Tidied up the documentation.	403	
1994-05-14 ltoutenc.dtx v1.5e	1994-05-16 ltoutenc.dtx v1.5h	
General: Replaced <code>\ENC@cmd</code> by <code>\ENC-cmd</code>	403	
1994-05-15 ltfssbas.dtx v2.1o	1994-05-16 ltoutenc.dtx v1.5i	
General: encoding cmd changed to <code>enc-cmd</code>	459	
1994-05-16 fontdef.dtx v2.1g	1994-05-16 ltoutput.dtx v1.0q	
General: Removed <code>\DeclareFontEncoding</code> for ot1 and t1 and input .def files instead . . .	600	
1994-05-16 ltalloc.dtx v1.1a	<code>\@writesetup</code> : Changed setting of accents (FMi): with the new encoding setup they can use <code>\let</code> . It could also use the new internal commands?	1012
General: (ASAJ) Split from ltinit.dtx.	322	
1994-05-16 ltcntrl.dtx v1.0a	General: Changed setting of accents (FMi).	984
General: (ASAJ) Split from ltinit.dtx.	324	
1994-05-16 ltdefns.dtx v1.1a	1994-05-16 ltpar.dtx v1.1a	
General: (ASAJ) Split from ltinit.dtx.	80	
1994-05-16 lterror.dtx v1.1a	General: (ASAJ) Split from ltinit.dtx.	339
General: (ASAJ) Completely new error interface.	328	
(ASAJ) Split from ltinit.dtx.	328	
1994-05-16 ltfinal.dtx v1.0i	1994-05-16 lplain.dtx v1.0h	
General: moved output enc stuff to lffonts	1076	
	General: Comment out encoding specific commands	32
	Remove <code>\@acci</code> and friends again .	33
	Remove unnecessary def for <code>\item</code> .	32
	<code>\loop</code> : Use Kabelschacht method .	30
	<code>\m@th</code> : Remove unnecessary space .	32

1994-05-16	ltspace.dtx v1.1a	Replaced \defaultencoding with \encodingdefault.	403
	General: (ASAJ) Split from ltinit.dtx.	360	
1994-05-17	ltclass.dtx v1.0e		
	\useoption: Execute option after removing from list, not before . . .	893	
1994-05-17	ltdefns.dtx 1.1b		
	General: (ASAJ) Added the \protect@... commands.	90	
1994-05-17	ltdefns.dtx v1.1b		
	General: (ASAJ) Added definitions for protect.	80	
	(ASAJ) Removed warnings and logging to lterror.dtx.	80	
	Added the discussion of protected commands, defined the values that \protect should have.	90	
1994-05-17	ltdefns.dtx v1.1c		
	General: (ASAJ) Redid definitions for protect.	80	
1994-05-17	lterror.dtx v1.1b		
	General: (ASAJ) Moved error stuff from ltdefns.dtx.	328	
1994-05-17	ltfssini.dtx v2.1n		
	\copyright: Really add extra braces	593	
	\nfss@text: Added braces to allow use in subscripts	593	
1994-05-17	ltmath.dtx v1.0i		
	General: Replaced \let by \gdef, for indirect definition.	700	
1994-05-17	ltoutenc.dtx v1.5j		
	General: Added braces to \pounds so it works as a subscript.	403	
1994-05-18	ltdefns.dtx 1.1c		
	General: (ASAJ) Renamed the commands, and removed one which is no longer needed.	90	
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.	90	
1994-05-18	ltfinal.dtx v0.1j		
	General: Corrected the lccode for d-bar.	1060	
1994-05-18	ltlogos.dtx v1.1b		
	General: (ASAJ) Added the TeX logo.	379	
	(ASAJ) Made the LATEX 2 ε logo use the text font '2' rather than the math font '2'.	379	
1994-05-18	ltoutenc.dtx v1.5k		
	General: Made dotted-i produce 'i'. . .	403	
	Removed braces from \pounds and \dollar.	403	
1994-05-19	ltbibl.dtx v1.1a		
	General: Initial version of ltbibl.dtx, split from ltidxbib.dtx	842	
1994-05-19	ltcounts.dtx v1.1a		
	General: Extracted file from ltcntlen.	449	
1994-05-19	ltdefns.dtx v1.1d		
	General: (RmS) Added definitions for \namedef and \nameuse again. . .	80	
1994-05-19	ltfinal.dtx v0.1k		
	General: Removed \makeat...	1060	
1994-05-19	ltidxglo.dtx v1.1a		
	General: Initial version of ltidxglo.dtx, split from ltidxbib.dtx	839	
1994-05-19	ltlength.dtx v1.1a		
	General: Extract file ltlength from ltcntlen.	457	
1994-05-19	ltpageno.dtx v1.1a		
	General: Extract file ltpageno from ltcntlen.	663	
1994-05-19	lplain.dtx v0.1k ltfinal		
	\showoutput: used \maxdimen not 99999	34	
	\showoverfull: used \one not 1 . . .	34	
1994-05-19	ltxref.dtx v1.1a		
	General: Extract file ltxref from ltcntlen.	664	
1994-05-20	ltdefns.dtx v1.1e		
	General: Changed command name from \checkcommand to \CheckCommand.	80	
	\CheckCommand: Changed name from \checkcommand to \CheckCommand.	88	
1994-05-20	lterror.dtx v1.1c		
	General: (ASAJ) Added \@lateinfo@no@line.	328	
	(ASAJ) Added missing full stops. .	328	
	(ASAJ) Fixed a bug with \inmatherr.	328	
1994-05-20	ltfinal.dtx v0.1l		
	General: Use new font warning commands	1067	
1994-05-20	ltfloat.dtx v1.0h		
	\endfloatbox: Restore outer value of \nobreak switch.	825	
	\outer@nobreak: Macro added: default is to do nothing.	825	
1994-05-20	ltfntcmd.dtx v3.3h		
	General: Use new error commands .	622	
1994-05-20	ltfssbas.dtx v2.1q		
	General: Use new error commands .	459	

1994-05-20 lfsstrc.dtx v2.3i		(ASAJ) Replaced \\ and tilde by \\MessageBreak and \\space.	328
General: Use new error command names	508		
1994-05-20 ltmiscen.dtx v1.0l		(ASAJ) Replaced \\@generic@message and \\@generic@error by \\GenericError, \\GenericWarning and \\GenericInfo.	328
\\@writefile: Added correct setting of \\protect.	677		
1994-05-20 ltmiscen.dtx v1.0m		(ASAJ) Replaces \\string by \\protect in some messages.	328
General: Use new warning commands	671		
1994-05-20 ltoutput.dtx v1.0s		1994-05-22 lterror.dtx v1.2d	
\\@writesetup: Added setting of \\protect during \\shipout.	1011	\\GenericError: (DPC) Alternative version added for old TeXs	329
General: Added setting of \\protect during \\shipout.	984	(DPC) New version using long command name.	329
1994-05-20 ltpage.dtx v1.0d		1994-05-22 ltfloat.dtx v1.0i	
\\markright: Changed setting for \\protect.	870	General: Use new warning commands	816
1994-05-20 ltsect.dtx v1.0c		1994-05-22 ltoutput.dtx v1.0t	
General: Correct setting of \\protect.	813	General: Changed warnings and infos to new commands.	984
\\addcontentsline: Correct setting of \\protect.	812	1994-05-22 ltpictur.dtx v0.1e	
1994-05-21 ltbibl.dtx v1.1b		General: Use new warning cmd	767
General: Use new warning commands	842	1994-05-23 ltclass.dtx v1.0h	
1994-05-21 lterror.dtx v1.1d		\\NeedsTeXFormat: Don't stop completely when format is wrong	896
General: (ASAJ) Made the error commands robust.	328	\\usepackage: Remove argument if possible	896
1994-05-21 ltfiles.dtx v1.0h		1994-05-23 ltdirchk.dtx v1.0f	
General: Use new error commands	380	General: Document \\@TeXversion	1
1994-05-21 ltlists.dtx v1.0f		1994-05-23 lfsstrc.dtx v2.3j	
General: Use new error commands	709	General: Removed def of \\f@warn@break	526
1994-05-21 ltmiscen.dtx v1.0n		1994-05-23 ltoutput.dtx v1.0u	
General: Use new error commands	671	\\@activechar@info: Added \\MessageBreak	1011
1994-05-21 ltsect.dtx v1.0d		\\@writesetup: Changed resetting of \\protect after shipout to use \\aftergroup	1011
General: Use new error commands	803	General: Added \\MessageBreak.	984
1994-05-21 ltab.dtx v1.0f		Changed resetting of \\protect after shipout.	984
General: Use new error commands	742	1994-05-24 lterror.dtx v1.2e	
1994-05-21 ltxref.dtx v1.1b		\\@latex@info@no@line: Macro added	332
General: Use new warning commands	664	1994-05-24 lterror.dtx v1.2f	
\\newlabel: Use new warning commands	667	General: (DPC) wrap long lines	328
1994-05-22 ltclass.dtx v1.0f		1994-05-24 lfntcmd.dtx v3.3i	
General: Use new warning and error commands	874	General: Tidying and typos fixed	622
1994-05-22 ltdefns.dtx v1.1f		1994-05-24 ltmiscen.dtx v1.0q	
General: Use new warning and error cmds	80	\\currenvline: Use \\empty as outer default	682
1994-05-22 lterror.dtx v1.1e		1994-05-25 ltdirchk.dtx v1.0g	
General: (ASAJ) Replaced bgroup by begingroup in error messages, to stop extra mathords creeping into math mode.	328	\\filename@parse: Mac parser had " typo for :	12
1994-05-22 lterror.dtx v1.2a			
General: (ASAJ) Made \\GenericError, \\GenericWarning and \\GenericInfo robust.	328		

1994-05-25 lfntcmd.dtx v3.3j		1994-06-01 ltlogos.dtx v1.1d
General: Insertion of <code>\aftergroups</code> to implement <code>\nocorr</code> moved to the end of the group	622	<code>\LaTeX:</code> Add <code>\m@th</code> to force math size calculations
<code>\check@icr:</code> Macros added	626	1994-06-01 ltoutput.dtx v1.0w
<code>\check@nocorr@:</code> Insertion of <code>\aftergroups</code> moved and defaults set up for efficiency	626	General: Tidied up typesetting.
<code>\DeclareTextFontCommand:</code> <code>\expandafter</code> inserted	624	1994-06-08 ltfinal.dtx v1.0m
Insertion of <code>\aftergroups</code> moved	624	General: Add patch file system
1994-05-25 ltoutput.dtx v1.0v		1994-06-09 ltfinal.dtx v1.0n
General: Extra documentation.	984	General: For <code>\TeX2</code> , do not set codes for higher half of character table.
1994-05-25 ltsect.dtx v1.0e		1994-06-09 lfntcmd.dtx v3.3k
<code>\@dottedtocline:</code> Put braces around argument 4 (the actual toc entry) to avoid font (and possibly other) changes leaking out to the leaders.	814	General: Tidying and typos fixed in documentation
1994-05-25 ltthm.dtx v1.0c		1994-06-18 lfntcmd.dtx v3.3l
General: Modify documentation . . .	799	General: Added check for empty text
1994-05-25 ltvers.dtx v1.0d		<code>\check@nocorr@:</code> Added check for empty text
General: Remove PRELIMINARY TEST RELEASE from startup banner (spring is here)	38	1994-06-22 lfntcmd.dtx v3.3m
1994-05-25 ltxref.dtx v1.1c		General: Removed space from <code>\nfss@text</code>
General: Modify documentation . . .	664	Renamed <code>\check@nocorr</code>
1994-05-26 ltfiles.dtx <code>\LaTeX2e</code>		<code>\check@nocorr@:</code> Renamed <code>\check@nocorr</code> to <code>\text@command</code> to improve <code>\long</code> error message .
<code>\@missingfileerror:</code> Modify message format	399	<code>\DeclareTextFontCommand:</code> Removed space from <code>\nfss@text</code>
1994-05-26 ltlogos.dtx v1.1c		1994-06-22 ltmath.dtx v1.2t classes
General: Remove <code>\SLiTeX</code> logo	379	<code>\mathindent:</code> Set <code>\mathindent</code> at the end of the class instead of at begin document
1994-05-26 ltmiscen.dtx v1.0r		1994-07-20 ltlogos.dtx v1.1e
General: <code>\literal</code> removed	692	<code>\LaTeX:</code> Save a few tokens
1994-05-26 ltpplain.dtx v1.1m		<code>\LaTeXe:</code> Save a few tokens
<code>\iterate:</code> (CAR) added <code>\long</code>	30	1994-07-20 ltpage.dtx v1.0h
<code>\underbar:</code> (FMI) changed to use box <code>\tw@</code>	32	<code>\sloppy:</code> Save a few tokens
1994-05-26 ltpplain.dtx v1.1p		1994-09-16 ltfssbas.dtx v2.1s
<code>\underbar:</code> (DPC) changed to use <code>\sbox</code>	32	<code>\nfss@catcodes:</code> Reset [and] as well, just in case
1994-05-29 ltfssdcl.dtx v2.1j		1994-10-07 ltoutenc.dtx v1.5l
General: Use new error commands .	538	General: Moved the ogonek accent. .
1994-05-31 ltfinal.dtx v1.0n		1994-10-11 ltdirchk.dtx v1.0h
General: Renamed <code>lthyphen.*</code> to <code>lthyphen.*.</code>	1060	<code>\@TeXversion:</code> Check for <code>\TeX3.14</code> . .
1994-06-01 ltboxes.dtx v1.0i		General: Modify all of <code>ltxcheck</code> again
<code>\@framebox:</code> Macro added.	732	1994-10-12 ltsect.dtx v1.0f
<code>\@ifframebox:</code> New version, so <code>\width</code> is correct in <code>\framebox</code>	732	General: Doc. typos
<code>\fbox:</code> New version, using <code>\@framebox</code>	731	1994-10-14 fontdef.dtx v2.2a
<code>\framebox:</code> New version, so <code>\width</code> is correct in <code>\framebox</code>	732	General: New coding
		1994-10-14 ltfssini.dtx v2.2a
		General: New coding for <code>cfg</code> files . .
		1994-10-14 ltmiscen.dtx v1.0s
		General: Move math to other file . .
		1994-10-14 ltpplain.dtx v1.1a
		General: Moved code to other files. .

1994-10-15 ltfssbas.dtx v2.1t		1994-10-25 ltboxes.dtx v1.0l	
\extract@alph@from@version: Warn if math alpha is used outside math 482		\@isavepicbox: missing percent (moved from ltpatch) 730	
1994-10-18 ltboxes.dtx v1.0j		1994-10-25 ltdefns.dtx v1.2b	
\Qframeb@x: \leavevmode added .. 732		General: Documentation improvements 80	
\Qiframebox: \leavevmode moved to \Qframeb@x 732		1994-10-25 ltoutenc.dtx 1.6a	
\Qparboxto: Macro added to remove misuse of \Qempty 734		General: Added \textdollar, \textlbrace, \textrbrace, \textsterling, \textunderline. 421	
General: stuff from ltpatch done ... 726		Removed \textlbrace, \textrbrace, \textunderline to give them their proper names. .. 421	
\fbox: \long added 731		1994-10-25 ltoutenc.dtx v1.6a	
\mbox: \long added 727		General: Added \ProvideTextCommand, \UseTextSymbol, \UseTextAccent, \DeclareTextSymbolDefault, \DeclareTextAccentDefault, \DeclareTextCommandDefault, and \ProvideTextCommandDefault. . 403	
\sbox: \long added 730		Added the \Provide commands, and the default definitions. 407	
1994-10-18 ltclass.dtx v1.0j		Added the defaults. 415	
General: Move \listfiles to ltfiles.dtx 874		Added the files OT1enc.def, T1enc.def and OMSenc.def. 415	
1994-10-18 ltdefns.dtx v1.2a		Added the OMS encoding. 427	
\Qstar@or@long: macro added 83		1994-10-27 ltoutenc.dtx 1.6b	
General: Add extra test for \endgraf 80		General: Added \textasciicircum	
Add star-forms for all commands . 80		\textasciitilde \textbackslash \textbar \textbraceleft \textbraceright \textcompwordmark \textemdash \textendash \textexclamdown \textgreater \texthyphenchar \texthyphen \textless \textquestiondown \textquotedblleft \textquotedblright \textquotedbl \textquotelleft \textquoteright \textunderscore \textvisiblespace 421	
\renew@environment: reset end command 87		Added: \textemdash \textendash \textexclamdown \texthyphenchar \texthyphen \textquestiondown \textquotedblleft \textquotedblright \textquoteright \textunderscore \textvisiblespace 419	
1994-10-18 ltfiles.dtx v1.0i		1994-10-27 ltoutenc.dtx v1.5d	
\listfiles: code moved here from ltclass 401		General: Rewrote	
1994-10-18 ltoutenc.dtx v1.5l			
General: Added new definitions of \patterns and \hyphenation. .. 415			
1994-10-18 ltoutenc.dtx v1.5m			
General: Added new definitions of \patterns and \hyphenation. .. 403			
1994-10-18 ltsect.dtx v1.0g			
\Qdottedtocline: Added \normalcolor for page number . 814			
General: Added \normalcolor 803			
1994-10-19 ltfssbas.dtx v2.1t			
\DeclareFontEncoding: Add missing \relax. 463			
1994-10-23 lfsstrc.dtx v23.k			
\every@math@size: Renamed to \every@math@size 517			
1994-10-23 ltmath.dtx v1.0l			
\eqnnum: Added \normalcolor since \eqno introduces a subgroup of the displayed math group 702			
\ensuremath: Remove extra braces: but see p 168 of Leslie's book .. 705			
1994-10-24 ltboxes.dtx v1.0k			
\fbox: Inner braces added (to fix latex/1061) 731			
1994-10-25 fontdef.dtx v2.2c			
General: Added OMSenc.def 600			

\DeclareTextSymbol to define its argument to use the current encoding by default, to fit with \DeclareTextCommand.	407	Moved math commands here from ltmath.	419
1994-10-27 ltoutenc.dtx v1.6b		Removed \textregistered.	417
General: Added \textbackslash.	427	Rewrote \copyright to use \textcircled.	417
Added more defaults for OT1.	415	1994-10-31 fontdef.dtx v2.2d	
Removed the enc.def files	403	General: Added OMLenc.def	600
Removed the files OT1enc.def, T1enc.def and OMScn.def.	415	1994-10-31 fontdef.dtx v2.2e	
Renamed \textlbrace to \textbraceleft and \textrbrace to \textbraceright.	427	General: ... and moved further down	600
1994-10-29 ltmath.dtx 1.0m		1994-10-31 ltfloat.dtx v1.1a	
General: ASAJ: Added \DeclareMathOperator.	693	\@dblflo: Major changes since two-column and one-column cases merged	819
ASAJ: Tidied up documentation.	701	\@dblfset: Macro added	819
1994-10-29 ltmath.dtx v1.0m		Major changes to parameter parsing, setting of local variables, etc; two-column and one-column cases merged; space hacks moved	819
General: ASAJ: Added \mathellipsis, \mathdollar and \mathsterling.	700	\@endfloatbox: (DPC/CAR) Extra box added to remove colour resetting from vmode	825
ASAJ: Removed \dag, \ddag.	700	\@floatboxreset: Macro added	823
ASAJ: Renamed \S and \P to \mathsection and \mathparagraph and made them \mathchardefs.	700	\@footnotetext: (DPC/CAR) Move colour setting to output routine	835
1994-10-29 ltoutenc.dtx v1.6c		\@savemarbox: (DPC/CAR) Extra box added for colour	829
General: Added commands like \dots for use in text and math.	415	\@setfps: Macro added	820
Renamed \P, \S, \dag and \ddag to \textparagraph, \textsection, \textdagger and \textdaggerdbl.	403	\@xdblfloat: Macros removed: \@dbfl, \@xdblfloat	825
1994-10-30 ltdefns.dtx v1.2c		\@xfloat: (DPC/CAR) Extra box added to remove colour resetting from vmode	821
\@onelvel@sanitize: Macro added	109	Major changes, removing setting of local variables, space hacks etc; two-column and one-column cases merged	820
General: (CAR)\@onelvel@sanitize added	80	Reset hook added	821
1994-10-30 ltdefns.dtx v1.2f		\@xmpar: (DPC/CAR) Extra box added since needed for floats	830
General: (DPC)\newwrite's moved to lfiles	80	\fps@db1: Macro added	820
1994-10-30 ltmath.dtx v1.0n		1994-10-31 ltoutput.dtx v1.1a	
General: ASAJ: Moved the new commands to ltoutenc.	700	\@makecol: (DPC/CAR) Colour resetting moved to here	1008
1994-10-30 ltoutenc.dtx v1.6d		\@topnewpage: (DPC/CAR) Extra box added to remove colour resetting from vmode	1000
General: Added \DeclareTextCompositeCommand.	403	(DPC/CAR) Use \color@begingroup for colour	1000
Added \textcircled.	403, 417, 427	(DPC/CAR) Use \normalcolor	1000
Added \t.	417	1994-11-02 ltoutenc.dtx v1.6d	
Added math commands.	403	General: Wrapped lines longer than 70 characters.	403
Added OML encoding.	403, 417		
Added the OML encoding.	428		
Made \textless and \textgreater come from OML.	417		

1994-11-03 ltclass.dtx v1.0k		1994-11-04 ltpage.dtx v1.0e
General: Move \@missingfileerror to ltfiles 878		\markright: Added \unexpandable@protect. ASAJ. 870
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. 807
1994-11-03 ltfiles.dtx v1.0j		General: (ASAJ) Added \protected@xdef to \thanks. ... 803
\@missingfileerror: Move here from ltclass 399		1994-11-04 ltsect.dtx v1.0h
1994-11-04 ltboxes.dtx v1.0m		General: Added \protected@write to \addtocontents. ASAJ. 813
\@mpfootnotetext: Added \protected@edef. ASAJ. 737		\addcontentsline: Added \protected@write to \addcontentsline. ASAJ. 812
1994-11-04 ltdefns.dtx v1.2e		1994-11-04 ltab.dtx v1.0h
General: Added \set@display@protect to \typeout. ASAJ. 80		\@mkpream: (ASAJ) Added \unexpandable@protect to \@mkpream. 761
Added commands for setting and restoring \protect. ASAJ. 92		\multicolumn: (ASAJ) added \set@typeset@protect. 757
Rewrote protected short commands using \x@protect. ASAJ. 90		1994-11-04 ltxref.dtx v1.1d
1994-11-04 lterror.dtx v1.2g		\label: (ASAJ) Added \protected@edef 668
General: Added \set@display@protect to \Generic* commands. ASAJ. ... 328		(ASAJ) Added \protected@write 668
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. 387		\@mpfootnotetext: Color resetting for footnotes moved to endminipage: as for main page. 737
\protected@write: Macro added ASAJ. 387		\color@endbox: macro added for color support 729
1994-11-04 ltfloat.dtx v1.1b		\color@hbox: macro added for color support 729
\@footnotetext: (ASAJ) Added \protected@edef. 835		\endminipage: Color resetting for footnotes moved to here: as for main page. 737
\ffootnotemark: Added \protected@xdef to \ffootnotemark. 836		1994-11-05 ltboxes.dtx v1.0o
1994-11-04 ltidxglo.dtx v1.1b		\@mpfootnotetext: Color groups restored here. 737
\@wrglossary: Added \protected@write to \@wrglossary. 841		1994-11-05 ltfloat.dtx v1.1c
\@wrindex: Added \protected@write to \@wrindex. 840		\@dblflset: Add compatibility with old version of \@xfloat. 819
General: Removed \if@filesw from \makeindex. 839		\@endfloatbox: Use new \color@hbox concept. 825
\makeglossary: Removed \if@filesw from \makeglossary. 840		\@footnotetext: Removed \normalcolor (again) 835
1994-11-04 ltmiscen.dtx v1.0t		\@savemarbox: Use new \color@hbox concept. 829
\@writefile: Removed setting of \protect. ASAJ. 677		\@setfps: Add compatibility with old version of \@xfloat. 820
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 820 Use new \color@hbox concept. ... 821
General: Added _ 418		
Added \mathunderscore. 419		

\@xypar: Use new \color@hbox concept.	830	(DPC) reduce save stack usage latex/1742	485
1994-11-05 ltoutenc.dtx v1.6g General: Added setting of \@typeset@protect to \patterns and \hyphenation.	415	1994-11-10 ltbibl.dtx v1.1c General: Fix \nocite{*}	842
1994-11-05 ltoutput.dtx v1.1b \@topnewpage: Use new \color@hbox concept.	1000	\nocite: Fix \nocite{*}	845
\@writesetup: Change protect settings for new-style, protect-free aux-files.	1011	1994-11-10 ltmath.dtx v1.2v classes eqnarray: Added value of \parskip to \abovedisplayskip to compensate for negative \topsep	708
1994-11-05 ltoutput.dtx v1.1c \@begindvi: Added macro	1015	1994-11-10 ltoutput.dtx v1.1e \@writesetup: Modify \protect setting	1011
\@begindvibox: Added macro	996	1994-11-10 lplain.dtx v1.1b General: (CAR) added patch to \loop.	15
\@writesetup: Add new \AtBeginDvi concept	1011	\iterate: (CAR) added extra \relax	30
\AtBeginDvi: Added macro	996	1994-11-11 lspace.dtx v1.2a \: (DPC) Make robust	364
1994-11-06 ltfsbas.dtx v2.1u \cf@encoding: New macro	469	1994-11-12 lfntcmd.dtx v3.3o \normalsize: Added \MessageBreak	630
\DeclareFixedFont: Renamed \every@size to \every@math@size	461	1994-11-12 llists.dtx v1.2b lspace \endtrivlist: Changed order of tests to make \noitemerror correct: end of an era.	718
1994-11-06 ltfsini.dtx v2.2b \@setsizes: Use \@typeset@protect	592	1994-11-12 ltmiscen.dtx v1.0u center: Changed end macro to \def: safer and consistent	684
1994-11-06 lfsstrc.dtx v2.3k \glb@currsize: New implementation	516	flushleft: Changed end macro to \def: safer and consistent	685
\try@simples: New implementation	527	flushright: Changed end macro to \def: safer and consistent	686
\try@size@substitution: New implementation	527	1994-11-12 lplain.dtx v1.1c General: Comment out more encoding specific commands	32
\tryis@simple: New implementation	527	1994-11-12 lspace.dtx v1.2b \addpenalty: Corrected error message	372
1994-11-07 fontdef.dtx v2.2f General: (DPC) Add \DeclareMathSizes declarations	605	\addvspace: Corrected error message	371
(DPC) Updated to use \ProvidesFile	600	1994-11-13 lspace.dtx v1.2c \addpenalty: Recorrected error message	372
1994-11-07 ltfiles.dtx v1.0m \document: Renamed \every@size to \every@math@size	383	\addvspace: Recorrected error message	371
1994-11-07 lplain.dtx v1.0l \@unused: move here from ltdefns, remove duplicate \mainaux	24	1994-11-14 ltoutput.dtx v1.1f \@begindvi: Use normal box register: why a box?	1015
1994-11-07 preload.dtx v2.1e General: (DPC) Updated to use \ProvidesFile	619	\@begindvibox: Use normal box register: why a box?	996
1994-11-09 ltboxes.dtx v1.0p \@finalstrut: Revert \finalstrut to 2.09 equivalent (from ltpatch)	740	\@writesetup: Modify new \AtBeginDvi concept	1011
General: more color changes...	726	General: Removed old definition of \@testfp.	984
1994-11-09 ltfsbas.dtx v2.1v \vpt: (DPC) macros added, from setsizes.dtx	485	\\: (DPC) Macro modified	364

1994-11-14 lttab.dtx v1.0i \tabularnewline: (DPC) Macro added 756	1994-11-18 ltfsdcl.dtx v2.1m \DeclareMathDelimiter: (DPC) \xpandafter instead of \next . 562
1994-11-16 fontdef.dtx v2.2h General: (DPC) Removed \{ and \} 600	1994-11-18 ltfssrc.dtx v2.3m General: \next to \reserved@f . 508
1994-11-17 ltboxes.dtx v1.0q General: \tempa to \reserved@a .. 726	1994-11-18 ltmath.dtx v1.0p \phantom: (DPC) colour support .. 695 (DPC) use \xpandafter instead of \next 695
1994-11-17 ltclass.dtx v1.0l General: \tempa to \reserved@a .. 874	\prime@s: (DPC) use \let@token instead of \next and \xpandafter instead of \nxt .. 700
1994-11-17 ltcntrl.dtx v1.0b General: \tempa to \reserved@a .. 324	\smash: (DPC) colour support 696 (DPC) use \xpandafter instead of \next 696
1994-11-17 ltdefns.dtx v1.0g General: \tempa to \reserved@a .. 80	1994-11-21 ltfloat.dtx v1.1f \endfloatbox: Added reset of minipage flag 825
1994-11-17 ltdirchk.dtx v1.0j General: \tempa to \reserved@a .. 1	Corrected position of \outer@nobreak 825
1994-11-17 lterror.dtx v1.2h General: \tempa to \reserved@a .. 328	\marginparreset: Macro added ... 829
1994-11-17 ltfiles.dtx v1.0n General: \tempa to \reserved@a .. 380	\savemarbox: Added \setminipage etc 829
1994-11-17 ltfinal.dtx v1.0o General: \tempa to \reserved@a .. 1060	Added resetting of size and font .. 829
1994-11-17 ltfloat.dtx v1.1e General: \tempa to \reserved@a .. 816	Changed to \color@vbox 829
1994-11-17 lftntcmd.dtx v3.3p General: \tempa to \reserved@a .. 622	Use \setnobreak etc 829
1994-11-17 ltfsbas.dtx v2.1w General: \tempa to \reserved@a .. 459	\setminipage: Macro added 823
1994-11-17 ltfsdcl.dtx v2.1m General: \tempa to \reserved@a .. 538	\setnobreak: Macro added 823
1994-11-17 ltfssrc.dtx v2.3l General: \tempa to \reserved@a .. 508	\xfloat: Added \setminipage .. 821
1994-11-17 ltmath.dtx v1.0o General: \tempa to \reserved@a .. 693	Added resetting of size and font .. 821
1994-11-17 ltmiscen.dtx v1.0v General: \tempa to \reserved@a .. 671	Changed to \color@vbox so that large floats overflow at the bottom 821
1994-11-17 ltoutenc.dtx v1.6h General: (DPC) \tempa to \reserved@a 403	Missing percents reinserted after 4, 8: these are not numbers. 820
1994-11-17 ltoutput.dtx v1.1h General: \tempa to \reserved@a. . 984	Use \setnobreak 821
1994-11-17 ltpictur.dtx v1.0f General: \tempa to \reserved@a .. 767	\xympar: Changed to \color@vbox 830
1994-11-17 ltsect.dtx v1.0i General: \tempa to \reserved@a .. 803	1994-11-21 ltoutput.dtx v1.1i \addtocurcol: Added \if@nobreak test before float box 1026, 1029
1994-11-17 lttab.dtx v1.0j General: \tempa to \reserved@a .. 742	\specialoutput: Added \if@nobreak test 1004
1994-11-18 ltboxes.dtx v1.0r \color@vbox: macro added for color support 729	\topnewpage: Changed to \color@vbox 1000
1994-11-18 ltfinal.dtx v1.0n General: re-allow slots 127–255 . . 1072	1994-11-22 ltfsdcl.dtx v2.1o General: wrap long lines 538
1994-11-18 ltfsbas.dtx v2.1x General: (DPC) use \reserved@f not \next 459	1994-11-22 ltoutenc.dtx v1.6i General: Corrected \dots so that there's no kerning in monowidth fonts. 403
	Corrected typo with \mathunderscore. 403
	Fixed empty accents. Again. 403

1994-11-24 ltdefns.dtx v1.2h		argument containing lots of commands.	409
\@newenv: Added test for \endgraf	87		
1994-11-25 lplain.dtx v1.1f			
General: (DPC) Comment out lots of obsolete code	15		
1994-11-26 lfloat.dtx v1.1b			
\f footnote: (ASAJ) Added \protected@xdef.	835		
1994-11-28 lcntrl.dtx v1.0c			
General: Documentation improvements	324		
1994-11-30 lfiles.dtx v1.0o			
\@dofilelist: Macro added	402		
\listfiles: Use \@dofilelist	401		
\nofiles: There is no \@gobblethree.	387		
1994-11-30 ltfssbas.dtx v2.1y			
\f fontshape: Use \@current@cmd in \@enc@update. ASAJ.	468		
1994-11-30 ltmath.dtx 1.0q			
General: ASAJ: \DeclareMathOperator moved to AMSLATEX.	693		
1994-11-30 ltmiscen.dtx v1.0w			
\enddocument@kernel@warnings: (DPC) Do warnings even for \nofiles	673		
\enddocument: (DPC) Use \@dofilelist	673		
1994-11-30 ltoutenc.dtx 1.7a			
General: Redefined \a for the new scheme.	415		
1994-11-30 ltoutenc.dtx v1.6g			
General: Removed new definitions of \patterns and \hyphenation, since encoding-specific commands now expand in the mouth.	415		
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.	403		
Always load the enc.def file, so that the default encoding for the commands will change.	446		
Redefined \@changed@cmd to expand in the mouth.	407		
Removed \@changed@x@mouth since \@changed@x now expands in the mouth.	407		
Rewrote \@text@composite so it allows an empty argument, or an			
1994-12-01 ltfinal.dtx v1.0p			
General: Renamed lthyphen.* to hyphen.*.	1060		
1994-12-01 lthyphen.dtx v1.0g			
General: Rename lthyphen.ltx/cfg to hyphen.ltx/cfg	1058		
1994-12-01 lplain.dtx v1.1g			
General: (DPC) More doc changes . . .	15		
1994-12-02 fontdef.dtx v2.2i			
General: Commented out \ldots.			
ASAJ.	598		
1994-12-02 ltfsmini.dtx v2.2c			
\f copyright: \copyright is now in ltoutenc. ASAJ	593		
1994-12-02 ltlists.dtx v1.0e			
\f trivlist: RmS: Added check for looping	717		
1994-12-02 ltoutenc.dtx 1.7b			
General: Redefined \a properly. . . .	415		
1994-12-02 ltoutenc.dtx v1.7b			
General: Fixed a bug with \a.	403		
1994-12-04 lthyphen.dtx v1.0h			
General: Documentation edits for /1989	1058		
1994-12-05 ltoutenc.dtx v1.7c			
General: Added braces to \textcircled.	403		
1994-12-06 ltfsbas.dtx v2.1z			
\f DeclareFontEncoding: use \nfss@catcodes	463		
\vfss@catcodes: Added tab char as well	474		
1994-12-08 ltoutenc.dtx v1.7d			
General: Added \null and \sh@ft to \nb and \d.	403		
1994-12-08 ltab.dtx v1.0k			
\f array: Add \tabularnewline	755		
\vfarray: (DPC) Made it \relax	756		
1994-12-09 ltbibl.dtx v1.1d			
\f bibliographystyle: (DPC) Allow use in preamble.	845		
1994-12-10 lfloat.dtx v1.1g			
\f dblfloat: Old version reinstated temporarily	819		
\f dblfset: Macro removed temporarily	819		
\f setfps: Macro removed temporarily	820		
\f dblfloat: Macros reinserted temporarily	825		

\@xfloat: Old version reinstated temporarily	820	1995-04-21 ltclass.dtx v1.0m	\DeclareOption*: Made long /1498	890
Sanitization added temporarily	820	\endfilecontents: Close input check stream: latex/1487	907	
General: Some temps reinserted temporarily	816	1995-04-21 ltfinal.dtx v1.0q	General: Allow initial patch level 0	1076
\fps@dbl: Macro removed temporarily	820	1995-04-21 ltoutenc.dtx v1.7h	General: Added \null \k latex/1274	403
1994-12-10 lfntcmd.dtx v3.3q		1995-04-22 ltfiles.dtx v1.0p	\includeonly: Allow blanks in argument	388
\@@math@egroup: Don't read arguments	629	1995-04-22 ltmiscen.dtx v1.0x	General: Removed extra def of \gobble	671
\check@nocorr@: Use \space command for comparison	626	1995-04-23 ltsect.dtx v1.0j	\addcontentsline: Use \contentsline internally.	812
1994-12-10 ltssdcl.dtx v2.1p		1995-04-24 ltbibl.dtx v1.1e	\@citex: Add \mbox to undefined case: latex/1239.	844
\document@select@group: Surround with braces (add fourth arg)	544	1995-04-24 ltbibl.dtx v1.1f	\bibcite: Make \onlypreamble /1388.	843
\select@group: Surround with braces (add fourth arg)	541	1995-04-24 ltcntrl.dtx v1.0d	\@for: Don't expand second argument with \edef: /1317 (DPC)	327
1994-12-10 ltoutenc.dtx v1.7e		1995-04-24 ltoutput.dtx v1.1j	\fl@tracemessage: Do not add to kernel unless 'trace' specified	1043
General: Added documentation for the OML encoding.	403	1995-04-24 ltoutput.dtx v1.1l	\begindvibox: Add \vbox latex/1392	996
Replaced width with \width and ditto height in vrules.	403	1995-04-24 ltpage.dtx v1.0f	\writesetup: Reset \\ latex/1451 (DPC)	1012
1994-12-14 ltoutenc.dtx v1.7f		1995-04-24 ltpage.dtx v1.0f	\fussy: reset \emergencystretch latex/1344	873
General: Added braces to \copyright so it works unbraced in subscripts.	403	1995-04-24 ltplain.dtx v1.1h	\newlanguage: Remove remaining \outer declarations.	18
Added check for math mode in \changed@cmd.	403	1995-04-24 ltxref.dtx v1.1e	\newlabel: Make \onlypreamble for /1388.	667
Commented out \textasciicircum, \textasciitilde, \textbackslash, \textbar, \textgreater, \texthypenchar, \texthypen and \textless to save memory.	403	1995-04-25 ltdefns.dtx v1.2i	\check@c: Make \long for latex/1346	88
1995-01-12 ltmath.dtx v1.2y classes		\newenvironment: Parse arguments slowly but safely /1507	86	
\eqnnum: Added \normalcolor	706	1995-04-25 ltfiles.dtx v1.0q	\document: Removed execution of \every@size latex/1407	383
1995-03-03 ltoutenc.dtx 1.7g		1995-04-25 ltsect.dtx v1.0k	\dottedtocline: Added \hbox around dots.	814
General: Corrected an error in documentation referring to the tabular rather than the tabbing environment.	415			
1995-04-02 lfntcmd.dtx v3.3r				
\@@math@egroup: Read them again to be able to add \relax.	629			
1995-04-02 ltssdcl.dtx v2.1q				
\document@select@group: fix problem for pr/1275	544			
\select@group: fix problem for pr/1275	541			
\set@mathdelimiter: fix pr/1329	564			
1995-04-02 ltssini.dtx v2.2d				
\not@math@alphabet: add \noexpand to second part of message	591			

1995-04-27 ltboxes.dtx v1.0s		1995-05-07 ltplain.dtx v1.1j
\@frameb@x: Move \leavevmode for graphics/1512 732		General: Use \hb@xt@ 15
\@iframebox: Move \leavevmode for graphics/1512 732		General: Use \hb@xt@ 803
\@iirsbox: Move \leavevmode for graphics/1512 740		General: Use \hb@xt@ 742
\@irsbox: Move \leavevmode for graphics/1512 740		General: Use \hb@xt@ 844
\fbox: Move \leavevmode for graphics/1512 731		\bibitem: Removed unnecessary braces 843
\raisebox: Move \leavevmode for graphics/1512 739		\nocite: Use \@firstofone 845
1995-04-27 ltfiles.dtx v1.0r		1995-05-08 ltdefns.dtx v1.2k
\document: Added \global to support groups in hook 384		\typein: Use \@firstofone 82
1995-04-27 ltmiscen.dtx v1.0y		1995-05-08 ltdefns.dtx v1.2l
\enddocument: \@checkend moved after hook 672		\typein: Remove unnecessary braces 82
1995-04-27 ltplain.dtx v1.1i		Replace \def by \let 82
General: Move \hang and \textrtindent to latex209.def 32		1995-05-08 lfsstrc.dtx v2.3n
1995-04-29 ltcntrl.dtx v1.0e		\ifnot@nil: Use \@firstofone ... 521
General: Moved init of \protect to ltdefns.dtx 327		1995-05-11 fontdef.dtx v2.2j
Removed unused defs for \@setprotect and \@resetprotect 327		General: Updates to some plain macros 598
1995-04-29 ltdefns.dtx v1.2j		1995-05-12 ltclass.dtx v1.0n
\protect: Init \protect here 92		\DeclareOption*: Use \toks@ to remove need to double hash /1557 890
1995-04-29 ltpar.dtx v1.1b		1995-05-12 ltfloat.dtx v1.1h
General: (TO) Comments clean-up. 339		\@footnotemark: Add \nobreak to allow hyphenation. latex/1605 ... 837
1995-05-02 ltsect.dtx v1.0l		1995-05-12 ltpictur.dtx v1.0h
\@dottedtocline: Don't reset to \rmfamily 814		\picture: Macro added for latex/1355 769
1995-05-03 ltsect.dtx v1.0m		1995-05-12 ltvers.dtx v1.0e
General: TO: Promoted documentation to doc.sty standard 803		General: Add autoload docstrip guards 38
1995-05-06 ltsect.dtx 1.0n		Check for format older than 1 year 38
\@seccntformat: Use \quad instead of \hskip 809		1995-05-13 lfsstrc.dtx v2.3o
\@sect: Added \relax after \@seccntformat just in case ... 807		General: Use single hash mark in \DeclareOption 509
1995-05-07 ltboxes.dtx v1.0t		1995-05-16 ltfloat.dtx v1.1i
General: Use \hb@xt@ 726		\@makefnmark: Now use \textrttextsuperscript 834
1995-05-07 ltdefns.dtx v1.2k		\textsuperscript: Command added./pr1503 834
\hb@xt@: Macro added 81		\thefootnote: Streamlined parts of code. 833
1995-05-07 ltmath.dtx v1.0r		1995-05-17 ltboxes.dtx v1.0u
General: Use \hb@xt@ 693		\@irsbox: Removed surplus braces . 740
1995-05-07 ltoutput.dtx v1.1m		1995-05-17 ltdefns.dtx v1.0o
General: Use \hb@xt@ 984		\g@addto@macro: Make long for latex/1522 112
1995-05-07 ltpictur.dtx v1.0g		1995-05-17 ltlists.dtx v1.0g
General: Use \hb@xt@ 767		\@item: Removed surplus braces ... 721
		\@nbitem: Removed surplus braces . 722
		\enumerate: Use \thr@@ and remove surplus braces 723
		\itemize: Use \thr@@ 724

1995-05-18	ltfloat.dtx v1.1j		General: Moved definition of <code>\footins</code> and <code>\footnoterule</code> from ltplain.	833
	<code>\@makefnmark</code> : Added <code>\normalfont</code>	834		
	<code>\thempfootnote</code> : Added <code>\itshape</code>	833		
1995-05-19	ltpictur.dtx v1.1a			
	General: Support autoloading feature	767		
1995-05-20	ltcounts.dtx v1.1b			
	<code>\@definecounter</code> : Streamlined code	451		
	<code>\@fnsymbol</code> : Allowing both text and math	455		
	<code>\fnsymbol</code> : Streamlined code	454		
1995-05-20	ltcounts.dtx v1.1c			
	<code>\@definecounter</code> : And do it right	451		
1995-05-20	ltfloat.dtx v1.1k			
	<code>\@makefnmark</code> : Moved <code>\normalfont</code> back and use <code>\textsuperscript</code>	834		
	Moved <code>\normalfont</code> to <code>\textsuperscript</code>	834		
	<code>\textsuperscript</code> : Use <code>\normalfont</code>	834		
1995-05-21	ltfssdcl.dtx v2.1t			
	<code>\DeclareMathRadical</code> : Allow for undefined cs names	565		
1995-05-21	ltlists.dtx v1.0f			
	General: Moved to doc.sty standard	709		
1995-05-21	ltmath.dtx v1.0r			
	<code>\sqrt</code> : Use <code>\sqrtsign</code>	703		
	General: Remove <code>\mathhexbox</code> from this file	698		
	Update some plain macros	693		
	<code>\lefteqn</code> : Use <code>\rlap</code>	705		
	<code>\righteqn</code> : Use <code>\sqrtsign</code> instead of <code>\sqrt</code>	695		
1995-05-21	ltoutenc.dtx v1.7h			
	<code>\@inmathwarn</code> : Added several <code>\onlypreamble</code>	407		
1995-05-21	ltoutenc.dtx v1.7j			
	General: Updated some plain macros	419		
1995-05-21	ltplain.dtx v1.1j			
	General: Moved some code to other files	15		
1995-05-22	ltplain.dtx v1.1k			
	General: Definitions of <code>\footins</code> and <code>\footnoterule</code> moved to ltfloat.	34		
1995-05-22	lttab.dtx v1.1a			
	General: Support autoloading feature	742		
1995-05-23	ltfssini.dtx v2.2e			
	<code>\newfont</code> : Font assignment made local again	592		
1995-05-24	ltdefns.dtx v1.1l			
	<code>\newif</code> : (DPC) New implementation	87		
1995-05-24	ltdefns.dtx v1.2m			
	<code>\typein</code> : (DPC) New implementation	82		
1995-05-24	ltfloat.dtx v1.1l			
	<code>\textsuperscript</code> : Command added	834		
	General: Moved definition of <code>\footins</code> and <code>\footnoterule</code> from ltplain.	833		
	<code>\textsuperscript</code> : Use <code>\textsuperscript</code>	834		
1995-05-24	ltfssbas.dtx v3.0a			
	General: (DPC) Make file from previous file, fam.dtx 1995/05/20 v2.2d	459		
	<code>\mathgroup</code> : (DPC) No need to redefine <code>\newfam</code> as not outer	459		
1995-05-24	ltfsscmp.dtx v3.0a			
	General: (DPC) Make file from previous file, fam.dtx 1995/05/20 v2.2d	533		
1995-05-24	ltfssdcl.dtx v3.0a			
	General: (DPC) Make file from previous file, latint.dtx 1995/05/21 v2.1t	538		
1995-05-24	ltfssini.dtx v3.0a			
	General: (DPC) Make file from previous file, lfonts.dtx 1995/05/23 v2.2e	570		
	<code>\cal</code> : (DPC) Remove definition	596		
	<code>\mit</code> : (DPC) Remove definition	596		
1995-05-24	ltfsstrc.dtx v3.0a			
	General: (DPC) Make file from previous file, tracefnt 1995/05/16 v2.3o	508		
1995-05-24	ltfsstrc.dtx v3.0b			
	General: (DPC) Fix <code>\ProvidesFile</code> usage	508		
1995-05-25	ltclass.dtx v1.0p			
	<code>\endfilecontents</code> : Delete <code>\filecontents</code> after preamble	907		
1995-05-25	ltfilehook.dtx v1.0t			
	<code>\unquoting</code> : (CAR) added <code>\long</code>	943		
1995-05-25	ltfiles.dtx v1.0s			
	<code>\document</code> : Added check for <code>\topskip</code> zero	384		
1995-05-25	ltfiles.dtx v1.0t			
	<code>\iffileonpath</code> : (CAR) added <code>\long</code>	397		
	<code>\document</code> : Corrected typo	384		
	<code>\IfFileExistsC</code> : (CAR) added <code>\long</code>	395		
	<code>\nofiles</code> : (CAR) added <code>\long</code>	387		
	<code>\protected@write</code> : (CAR) added <code>\long</code>	387		
1995-05-25	ltfloat.dtx v1.1m			
	<code>\savemarbox</code> : (CAR) Reset settings moved to hook	829		
	<code>\xfloat</code> : (CAR) Reset settings moved to hook	821		

1995-05-25 ltlists.dtx v1.0i \endtrivlist: Macros moved from ltspace.dtx 718	1995-06-09 ltoutenc.dtx v1.7i \DeclareTextComposite: Rewrote \DeclareTextComposite to define the composite as a no-argument command rather than a two-argument command. 410
1995-05-25 ltmath.dtx v1.3c classes \eqnnum: replace \reset@font\rmfamily with \normalfont (PR 1578) 706	1995-06-11 ltspace.dtx v1.2g \restorerelax: (CAR) \relax added to stop silent eating of *. 378
1995-05-25 ltspace.dtx v1.2f \vbshack: (CAR) not used so 'removed'. 370	1995-06-13 ltfinal.dtx v1.0t General: Add patch level string more carefully 1076
\vspace: (CAR) \restorepar added to avoid possible infinite tail recursion caused by a typo in the argument. 373	Call \errorstopmode 1078
(CAR) macros modified to be more efficient 373	1995-06-13 ltpictur.dtx v1.1b General: Use \ProvidesFile in autoload 767
General: Macros moved to ltlists.dtx 360	1995-06-14 ltab.dtx v1.1b General: Use \ProvidesFile in autoload 742
1995-05-26 ltdefns.dtx v1.2n \gobblefour: (CAR) Added \longs 89	1995-06-15 lfssbas.dtx v3.0c General: (DPC) minor documentation changes 459
1995-05-26 ltmath.dtx v1.0s \eqnnum: Removed \rmfamily (PR 1578), replaced \reset@font with \normalfont 702	1995-06-15 lfsscjmp.dtx v3.0b General: (DPC) minor documentation edits 533
1995-05-26 ltpage.dtx v1.0g \psplain: removed \rmfamily (PR 1578) 870	1995-06-15 lfssdcl.dtx v3.0b General: (DPC) minor documentation changes 538
1995-05-27 lfssbas.dtx v3.0b \mathgroup: (FMi) But a need to define \newmathgroup 459	1995-06-19 ltbibl.dtx v1.1h \bibcite: Call \newl@bel so repeated keys produce better warning. 843
1995-06-05 fontdef.dtx v2.2k General: Moved math commands from ltoutenc.dtx. 616	1995-06-19 ltclass.dtx v1.0q \documentclass: Don't redefine \usepackage in compat mode for /1634 894
1995-06-05 ltfinal.dtx v1.0r General: Added \MakeUppercase and \MakeLowercase. 1060	1995-06-19 ltxref.dtx v1.1e \newlabel: Use \newl@bel to share code with \bibcite 667
1995-06-05 ltoutenc.dtx v1.7k \inmathwarn: Removed \protected@cmd and replaced with explicit \noexpand. 407	1995-06-28 lfssini.dtx v3.0b General: (DPC) Fix documentation typos 570
General: Allowed \ProvideTextCommandDefault after the preamble. 409	1995-06-28 ltmath.dtx v1.0t General: minor doc edits 693
Commented out \textless and \textgreater. 417	1995-07-02 ltplain.dtx v1.1n General: Removed surplus 'by' and '=' in various places 15
Moved math commands to fontdef.dtx. 419	\offinterlineskip: Replaced 1000 by \@m 31
Save some tokens in \textvisiblespace and \textunderscore. 417	\showoutput: Use \showoverfull to save space 34
1995-06-06 ltfinal.dtx v1.0s General: Made \MakeUppercase and \MakeLowercase brace their argument. 1060	\tracingall: Use \showoutput to save space 34

1995-07-03 ltdefns.dtx v1.2o		Shorten redefinition of \bincite and \newlabel	673	
	\set@typeset@protect: Use \@typeset@protect for init	92		
1995-07-03 lftntcmd.dtx v3.3s		\bincite: Remove \onlypreamble so still defined in new \enddocument	843	
	\t@st@ic: Use clean interface for jump	628		
1995-07-05 lftntcmd.dtx v3.3s		\newlabel: Remove \onlypreamble so still defined in new \enddocument	667	
	\t@st@ic: Renamed from \test@next .	628		
1995-07-05 ltspace.dtx v1.2h	\@newline: Use \break	366	1995-07-19 ltfssini.dtx v3.0d	
	\@no@pgbk: Macro replaces \pgbk and \@nopgbk	364	General: (DPC) TeX2 support	596
	\nopagebreak: Reimplemented both using \no@pgbk	363	1995-07-20 ltboxes.dtx v1.0v	
1995-07-09 ltcntrl.dtx v1.0f	\@iforloop: Reimplemented using Kabelschacht method	327	\@isavebox: Use \sbox	730
	\@iwhiledim: Reimplemented using Kabelschacht method	325	\@isavepicbox: Use \sbox	730
	\@iwhilenum: Reimplemented using Kabelschacht method	325	1995-07-21 ltoutput.dtx v1.1o	
	\@iwhilesw: Reimplemented using Kabelschacht method	325	\@writesetup: Command added	1011
	\@tfor: Reimplemented using Kabelschacht method	327	New, experimental, versions: need in-lining	1011
1995-07-09 ltlists.dtx v1.0j	\@break@tfor: Made long	327	1995-08-09 ltmath.dtx v1.0u	
	\@forloop: Made defs long	327	General: Added code for class options leqno and fleqn	706
	\@fornoop: Made defs long	327	1995-08-11 ltlength.dtx v1.1b	
	\@iforloop: Made defs long	327	General: Doc typos fixed for latex/753	457
	\@iwhiledim: Made defs long	325	1995-08-16 ltcntrl.dtx v1.0g	
	Removed \@whilenoop	325	\@break@tfor: Made long	327
	\@iwhilenum: Made defs long	325	\@forloop: Made defs long	327
	Removed \@whilenoop	325	\@fornoop: Made defs long	327
	\@iwhilesw: Removed \@whileswnoop	325	\@iforloop: Made defs long	327
	\@tfor: Made defs long	327	\@iwhiledim: Made defs long	325
1995-07-13 ltdefns.dtx v1.0p	1995-08-16 ltfssbas.dtx v3.0f		Removed \@whilenoop	325
	General: Updates to documentation .	80	\@iwhilenum: Made defs long	325
1995-07-13 ltfiles.dtx v1.0u	General: Updates to docu	380	Removed \@whilenoop	325
	\@defaultsubs: macro added	479	\@iwhilesw: Removed \@whileswnoop	325
1995-07-13 ltfssbas.dtx v3.0d	\@defaultsubs: macro added	479	\@tfor: Made defs long	327
	General: minor documentation changes	459	1995-08-16 ltfssbas.dtx v3.0f	
	\@wrong@fontshape: Change a macro not a switch to flag default font substitutions	478	General: Added autoload code	459
1995-07-13 ltmiscen.dtx v1.0z	\@centercr: Use \nobreak	683	1995-08-24 ltfsstrc.dtx v3.0c	
	\@enddocument@kernel@warnings: Use \@defaultsubs instead of switch .	674	General: Macro \gobble@font@spec removed	522
	\@writefile: Added missing percent and use \relax in the THEN case .	677	\tryis@simple:	529
	\@xobeysp: Use \nobreak	686	1995-08-25 ltoutput.dtx v1.1p	
	General: Improve Documentation .	671	General: Support autoloading feature (FMi).	984
	\enddocument: Set \setckpt to \gobbletwo instead of defining it by hand	672	1995-09-01 lterror.dtx v1.2i	
			General: Add autoload support	328
			1995-09-01 lplain.dtx v1.1m	
			\empty: Use \let to save space	28
			\I: Use \let to save space	28

1995-09-14 ltplain.dtx v1.1o General: Moved <code>\multispan</code> to <code>ltab.dtx</code>	15	1995-10-11 ltoutput.dtx v1.1r <code>\clearpage</code> : Added a check so that it does not lose the argument of <code>\twocolumn[...]</code>	997
1995-09-14 ltab.dtx v1.1c <code>\cline</code> : (DPC) New implementation	765	1995-10-16 ltbibl.dtx v1.1j <code>\cite</code> : (DPC) Make robust	843
1995-09-15 ltfssini.dtx v3.0e General: (DPC) Modify TeX2 message	596	1995-10-16 ltboxes.dtx v1.0w General: Clarify makebox description	726
1995-09-19 ltmiscen.dtx v1.1a <code>\verb</code> : Put <code>\@noligs</code> after <code>\verbatim@font</code> where it belongs.	691	1995-10-16 ltdefns.dtx v1.2u <code>\@ifstar</code> : (DPC) New implementation, for /1910	108
1995-10-01 ltfiles.dtx LaTeXe <code>\@addtofilelist</code> : Macro added . . .	401	<code>\new@command</code> : (DPC) Use <code>\@testopt</code> /1911	84
1995-10-02 ltdefns.dtx v1.2q <code>\@ifnch</code> : Use <code>\@let@token</code> for internal/924, save <code>\reserved@e</code> .	108	<code>\new@environment</code> : (DPC) Use <code>\@testopt</code> /1911	86
<code>\@ifnextchar</code> : Use <code>\@let@token</code> . . .	107	<code>\typein</code> : (DPC) Use <code>\@testopt</code> /1911	82
<code>\@protected@testopt</code> : Macro added .	85	1995-10-16 ltfssini.dtx v3.0f <code>\reset@font</code> : Added <code>\relax</code> after <code>\usefont</code> , as the latter eats up spaces.	593
<code>\@testopt</code> : Macro added	84	1995-10-16 ltmath.dtx v1.0y <code>\@yeqnqr</code> : (DPC) Use <code>\@testopt</code> /1911	704
<code>\@xargdef</code> : New implementation, using <code>\@test@opt</code>	84	<code>\sqrt</code> : (DPC) Make robust /1808 . .	703
1995-10-03 fontdef.dtx v2.21 General: <code>\@sqrt</code> from patch file for /1701	598	1995-10-16 ltspace.dtx v1.2j <code>\nolinebreak</code> : (DPC) Use <code>\@testopt</code> /1911	363
1995-10-03 ltdefns.dtx v1.2r <code>\typein</code> : Add missing <code>\@typein</code> for /1710 (from patch file)	82	<code>\nopagebreak</code> : (DPC) Use <code>\@testopt</code> /1911	363
1995-10-03 ltpictur.dtx v1.1e General: New autoload code	767	1995-10-16 lthm.dtx v1.0g General: Revert to previous <code>\newtheorem</code> behaviour	799
1995-10-04 ltfssbas.dtx v3.0g General: Modify autoload code	459	1995-10-17 ltclass.dtx v1.0r <code>\@providesfile</code> : Delay definition of <code>\ProvidesFile</code> till <code>ltfinal</code>	888
1995-10-04 ltfsstrc.dtx v3.0d General: (DPC) Modify autoload code	508	<code>\ProcessOptions*</code> : Reset <code>\CurrentOption</code> for graphics/1873	892
1995-10-04 ltab.dtx v1.1d General: Modify autoload support .	742	1995-10-17 ltdirchk.dtx v1.0l General: Modify initex version of <code>\ProvidesFile</code>	4
1995-10-06 ltfiles.dtx v1.0w <code>\@missingfileerror</code> : Autoload error	399	1995-10-17 ltfinal.dtx v1.0v <code>\@providesfile</code> : reset macro	1077
1995-10-09 lterror.dtx v1.2j General: Modify autoload support .	328	<code>\reserved@b</code> : reset here after the <code>\input</code> above	1077
1995-10-09 ltoutenc.dtx v1.7m <code>\@inmathwarn</code> : Autoload error . . .	408	1995-10-17 ltplain.dtx v1.1s <code>\reject</code> : Move <code>\supereject</code> to compat file	31
1995-10-10 ltfssbas.dtx v3.0h <code>\showhyphens</code> : Use <code>\normalfont</code> and make colour safe, and autoloadable	483	1995-10-17 ltab.dtx v1.1e <code>\cline</code> : (DPC) Use <code>\@multicnt</code> . . .	765
1995-10-10 ltfsdcl.dtx v3.0c <code>\non@alphe rr</code> : (DPC) autoload error message	542	<code>\@multispan</code> : (DPC) Macro added.	765
1995-10-10 ltplain.dtx v1.1r General: Autoload tracing code	15	1995-10-19 ltfinal.dtx v1.0w <code>\@filelist</code> : Move after <code>\reserved@a</code> setting:-)	1078
1995-10-10 ltthm.dtx v1.0f General: Make <code>\newtheorem</code> ‘only preamble’	799		

1995-10-20	ltbibl.dtx v1.1k		\@setref: Switch for refundefined renamed	666
	\@citex: Removed refundefined flag	844		
	\nocite: Removed refundefined flag	845	\if@multiplelabels: Macro removed	668
1995-10-20	ltclass.dtx v1.0s			
	\@begindocumenthook: Make setting conditional, for autoload version	905	1995-10-25 ltalloc.dtx v1.1b General: General doc improvements	322
1995-10-20	ltfssbas.dtx v3.0i			
	General: (DPC) Modify autoload code, change \undefined	459	1995-10-25 ltfloat.dtx v1.1n \endfloatbox: (CAR) macro added: to unify code for double and single versions	825
1995-10-20	ltfsstrc.dtx v3.0e		\enddblfloat: (CAR) unify code for double and single versions	824
	General: (DPC) Modify autoload code	508	\endfloat: (CAR) unify code for double and single versions	823
1995-10-22	ltfssbas.dtx v3.0j			
	General: (RmS) New size function macro \genb@sfcnt needs to be disabled at \document.	459	1995-10-25 ltdxglo.dtx v1.1d General: Doc cleanup	839
1995-10-22	ltfsstrc.dtx v3.0f			
	General: Added 'genb' and 'sgenb' size functions to support new DC font naming scheme.	508	1995-10-25 ltsect.dtx v1.0q \subparagraphmark: Use \let not \def to save space.	811
1995-10-23	lttab.dtx v1.1f			
	\@settab: (CAR) Ensure that \hightab increases by at most one	749	1995-10-27 ltpictur.dtx v1.1f General: Move initialization to kernel from autoload file	794
	\@startline: (CAR) Ensure that \nxttabmar is never larger than \hightab	747		
	\poptabs: (CAR) Ensure that \curtab is never larger than \hightab	750	1995-10-31 ltboxes.dtx v1.0x \finalstrut: Add \nobreak in horiz mode to allow hyphenation. internal/1931	740
	\tabbing: (CAR) Make \hightab consistently a local variable . . .	749		
1995-10-24	ltfiles.dtx v1.1a			
	\document: Removed multiplelabels switch	383	1995-11-01 fontdef.dtx v2.2m General: add \nfss@catcodes for internal/1932	601
	Removed refundefined switch . . .	383		
1995-10-24	ltfssbas.dtx v3.0k			
	\@defaultsubs: macro removed . .	479	1995-11-01 ltdirchk.dtx v1.0n General: Initialise \@addtolist to \gobble	4
	\wrong@fontshape: Make this code inline since it happens only here	478		
1995-10-24	ltmisen.dtx v1.1b			
	\enddocument@kernel@warnings: Changed logic for producing warning messages and removed switch	674	1995-11-01 ltfssini.dtx v3.0g General: (DPC) Switch meaning of \addtolist for cfg files . . .	596
	Use \@refundefined instead of switch	674		
1995-10-24	ltxref.dtx v1.1h			
	\@multiplelabels: Switch for multiplelabels removed	668	1995-11-02 ltfssbas.dtx v3.0n \wrong@fontshape: (DPC) Remove extra space with \string for latex/1676	478
	\newl@bel: Switch for multiplelabels replaced by inline code	667		
	\@refundefined: Switch for refundefined replaced	665	1995-11-02 ltoutenc.dtx v1.7n General: Changed internal name \a to \tabacckludge to protect against redefinition by malicious users. . .	415
			\doendpe: Enclosed \setbox0 assignment by a group so that it leaves the contents of box 0 intact.	719

1995-11-07 ltoutenc.dtx v1.7o	General: Added <code>\leavevmode</code> at start of <code>\c</code> , otherwise the output routine might be invoked within the macro.	419	1995-11-29 ltoutenc.dtx v1.7t	General: Added <code>\textasciicircum</code> , <code>\textasciitilde</code> , <code>\textbackslash</code> , <code>\textbar</code> , <code>\textgreater</code> and <code>\textless</code>	423
	Changed <code>\char32</code> to <code>\@xxxii</code> (two tokens less).	420		Added <code>\textasciicircum</code> , <code>\textasciitilde</code> , <code>\textregistered</code> and <code>\texttrademark</code>	417
	Replaced octal number 27 by decimal number 23 to protect against the quote character being active.	420		Added <code>\textbackslash</code> and <code>\textbar</code>	416, 427
	Replaced some 0's by <code>\z@</code> (faster).	420		Added <code>\textless</code> and <code>\textgreater</code>	417, 428
1995-11-10 ltoutput.dtx v1.1s	<code>\@shipoutsetup</code> : Command removed	1011	1995-12-01 ltoutenc.dtx v1.7u	General: Made <code>\\$S</code> a Default, rather than having the default point to the OT1 definition.	417
	<code>\@writeshop</code> : Command removed In-lined	1011	1995-12-04 ltspace.dtx v1.2k	<code>\nobreakspace</code> : (Macro added	376
1995-11-14 ltclass.dtx v1.0t	<code>\@unprocessedoptions</code> : Allow empty option	907	1995-12-04 ltspace.dtx v1.2l	<code>\@obeysp</code> : (braces added to definition of tilde	376
	<code>\@loadwithoptions</code> : macro added	895	1995-12-04 preload.dtx v2.4e	General: Ulrik Vieth. added 12pt OMS and OML preloads /1989 .	621
	<code>\LoadClassWithOptions</code> : macro added	895	1995-12-05 ltdefns.dtx 1.2w	<code>\@unexpandable@protect</code> : Removed <code>\unexpandable@noexpand</code> as never used. internal/1733	90
1995-11-17 ltfssbas.dtx v3.0m	<code>\@wrong@font@char</code> : (DPC) Macro added. latex/1676	479	1995-12-05 ltfiles.dtx v1.1c	<code>\document</code> : <code>\ignorespaces</code> added for latex/1933	384
	<code>\define@newfont</code> : Redefine <code>\typeout</code> latex/1676	473	1995-12-05 ltfloat.dtx v1.1n	<code>\@textsuperscript</code> : Use <code>\ensuremath</code> for latex/1984.	834
	<code>\wrong@fontshape</code> : Support <code>\@wrong@font@char</code> latex/1676 .	478	1995-12-05 ltoutenc.dtx v1.7v	<code>\@inmathwarn</code> : Changed <code>\TextSymbolUnavailable</code> text ..	408
1995-11-17 ltoutenc.dtx v1.7p	<code>\UseTextSymbol</code> : Support <code>\@wrong@font@char</code> latex/1676 .	412	1995-12-06 ltfssbas.dtx v3.00	<code>\nfss@catcodes</code> : Reset hat, for typeouts etc in fd files	474
1995-11-18 ltoutenc.dtx v1.7q	<code>\UseTextSymbol</code> : Modify message slightly	412	1995-12-07 ltbibl.dtx v1.1l	<code>\@citex</code> : Restored name of <code>\G@refundefinedtrue</code>	844
	General: Incorporate changed figures, as in plain.tex	614	1995-12-07 ltfloat.dtx v1.1m	<code>\@textsuperscript</code> : Move <code>\m@th</code> out of the <code>\ensuremath</code> for latex/1984.	834
1995-11-21 fontdef.dtx v2.2n	<code>\nfss@catcodes</code> : Reset hash, for definitions in fd files	474	1995-12-07 ltref.dtx v1.1i	<code>\@setref</code> : Switch for refundefined restored	666
	General: documentation fixes	816		<code>\G@refundefinedtrue</code> : Renamed (back) from <code>\G@refundefined</code> ..	665
1995-11-27 ltfssbas.dtx v3.0n	General: documentation fixes	508			
1995-11-28 ltfloat.dtx v1.1n	General: documentation fixes	407			
1995-11-28 lfsstrc.dtx v3.0g	doc fixes	403			
	General: documentation fixes	508			
1995-11-28 ltoutenc.dtx v1.7r	General: Added math mode checks to text commands.	407			
	Renamed <code>\@changed@x@err</code> to <code>\TextSymbolUnavailable</code>	403			
		407			

1995-12-11 ltoutenc.dtx v1.7w		1996-05-21 ltoutenc.dtx v1.7y	
General: Modified \copyright	417	General: Corrected error message (CAR)	446
1995-12-13 ltdefns.dtx 1.2x		1996-05-21 ltsect.dtx v1.0s	
\:-: Documentation changed.	110	\@sect: (DPC) Added extra braces for internal/2148	807
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	807
\@iffileonpath: Change argument handling to not require doubled hash. latex/2024	397	\@ssect: (DPC) Added extra braces for internal/2148	810
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	810
\makeglossary: Make no-op after use pr/2048	840	1996-05-23 ltoutenc.dtx v1.7z	
\makeindex: Make no-op after use pr/2048	840	\@strip@args: \expandafter added to match other changes for latex/2133	412
1996-01-20 ltspace.dtx v1.2m		\add@accent: macro added. latex/2133	409
\vspace: Made robust	373	\DeclareTextAccent: Reimplemented using \add@accent to save space latex/2133	409
1996-03-25 ltmath.dtx v1.1a		\DeclareTextCompositeCommand: Modified to cope with new \add@accent command: required removal of check for one argument-command	410
\@ensuredmath: Macro added for amslatex/2104	705	1996-05-24 ltoutput.dtx v1.1t	
\ensuremath: Reimplement for amslatex/2104	705	\@specialoutput: Check that \@colroom is less than \vsize, indicating that a float has been added	1002
1996-04-18 ltpage.dtx v1.0i		Cut-off point changed to 1.5\baselineskip	1002
General: Improve documentation	869	\@topnewpage: Cut-off point changed to 2.5\baselineskip	1001
1996-04-22 ltmiscen.dtx v1.1c		1996-05-25 ltoutput.dtx v1.1u	
General: Improve Documentation	671	\@specialoutput: Correct the above check	1002
1996-04-22 ltspace.dtx v1.2n		1996-06-03 ltmiscen.dtx v1.1d	
General: Documentation Improvements	360	\overline: Exchanged the following two code lines so that \dospecials cannot reset the category code of characters handled by \noligs.	687
1996-04-22 lttab.dtx v1.1g		General: Move setting of verbatim font and \noligs.	671
\@tabclassz: (DPC) Extra \hskip keeps tabcolsep in empty columns internal/2122	762	\verb: Put setting of verbatim font after \dospecials so that \dospecials cannot reset the	
1996-04-23 ltcnts.dtx v1.1d			
General: Documentation improvements	449		
1996-04-24 ltfiles.dtx v1.1e			
\document: (DPC) Reset \AtBeginDocument eg for latex/1297	383		
1996-05-08 lfsstrc.dtx v3.0h			
\math@egroup: Use \bgroup instead of \begingroup to match a kernel change made in 1994!!	520		
1996-05-09 lftntcmd.dtx v3.3t			
\check@icr: Default definitions added	626		
1996-05-17 fontdef.dtx v2.2o			
General: \@sqrt removed, at last	598, 614		
1996-05-17 ltfiles.dtx v1.1f			
\nofiles: added \write to \protected@write for latex/2146	387		
1996-05-18 ltoutenc.dtx v1.7x			
General: Produce error if encoding not found. pr/2054	446		

category code of characters handled by \noligs.	691	\nfss@catcodes: omit \relax as not needed	474
1996-06-10 ltboxes.dtx v1.0y \parboxto: (DPC) Changed \endgraf to \par	734	1996-07-26 ltfssdcl.dtx v3.0e \init@restore@version: Removed \ifrestore@version switch and replaced by \init@restore@version	474
1996-06-10 ltsect.dtx v1.0t \sect: (DPC) Changed \endgraf to \par	807	1996-07-26 lfsstrc.dtx v3.0i \init@restore@glb@settings: macro added replacing \if@inmath switch	542
1996-06-13 ltdirchk.dtx v1.0r General: documentation improvements mainly from internal/2174	1	1996-07-26 ltlsts.dtx v1.0l \item: Remove unnecessary \global before \minipage....	519
1996-06-14 lttab.dtx v1.1h \tabclassz: (DPC) Change both\z@skip to 1sp for latex/2160	762	\Remove unnecessary \global before \nobreak....	720
1996-06-22 ltspace.dtx v1.2o General: Documentation of problems added	360	1996-07-26 ltmath.dtx v1.1b General: Removed \global before \ignorestrue in various places.	721
1996-07-10 ltfinal.dtx v1.0y \toks: Free up memory from scratch registers /2213	1077	1996-07-26 ltmiscen.dtx v1.1e \ignorefalse: put \global into definition	693
1996-07-19 ltoutenc.dtx v1.8a \strip@args: Use char 0 not @ as carrier for \lowercase /2197 ...	412	\begin: remove \global before \ignore....	672
1996-07-26 ltboxes.dtx v1.0z \if@minipage: put \global into definition	735	\end: remove \global before \ignore....	679
1996-07-26 ltclass.dtx v1.0u \classoptionslist: made only preamble	879	\ignorespacesafterend: user level macro added	681
1996-07-26 ltdefns.dtx v1.2y \reargdef: third arg picked up by \yargdef	85	1996-07-26 ltoutput.dtx v1.1v \testfp: remove \global before \test....	672
1996-07-26 ltfloat.dtx v1.1n \endfloatbox: remove unnecessary \global before \minipage... .	825	\xtryfc: remove \global before \test....	1046
1996-07-26 ltfloat.dtx v1.1n \savemarbox: remove unnecessary \global before \minipage... .	829	\ztryfc: remove \global before \test....	1020
1996-07-26 ltfloat.dtx v1.1n \setminipage: remove unnecessary \global before \minipage... .	823	\clearpage: add number of missing percents	1021
1996-07-26 ltfloat.dtx v1.1n \setnobreak: remove unnecessary \global before \nobreak... .	823	General: put \global into definition remove \global before \test....	994
1996-07-26 ltfsbs.dtx v3.0p \DeclareMathSizes: use faster \if test	466	\clearpage: add number of missing percents	994
		\clearpage: add number of missing percents	997
		1996-07-26 ltplain.dtx v1.1t \sh@ft: replace \dimen\z@ by \dimen@	33
		1996-07-26 ltsect.dtx v1.0u \starttoc: removed \global before \nobreak....	812
		\xsect: Removed \global before \nobreak....	808
		1996-07-26 ltspace.dtx v1.2p \if@nobreak: put \global inside definition	366
		1996-07-27 ltfsbs.dtx v3.0q General: \if@inmath switch removed	472
		1996-07-27 ltspace.dtx v1.2q General: Further documentation of problems	360

1996-07-27 ltspace.dtx v1.2r		1996-10-21 ltab.dtx v1.1i	
General: Correct documentation of problems 360		\array: Use \set@typeset@protect 755	
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) 754	
\@xympar: Remove \global before \@ignore 830		\multicolumn: Make \multicolumn long (latex/2180) 757	
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) 749	
\@afterheading: Removed \global before \nobreak..... 810		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 721	
\@Esphack: Remove \global before \@ignore 369		\kern... changed to \setbox... 721	
1996-08-25 lfssbas.dtx v3.0r		Added setting of \clubpenalty and set \nobreakfalse only when necessary 721	
\nfss@catcodes: Reset the acute, grave and double quote chars as well 474		1996-10-23 ltdirchk.dtx v1.0t	
1996-09-21 ltoutput.dtx v1.1w		\xsect: Replaced \hskip... with \setbox... as used in \@afterheading 808	
\@writesetup: Added \@parboxrestore and made consequent deletions: wait for the howls of protest 1011		1996-10-24 ltboxes.dtx v1.1a	
1996-09-25 ltdirchk.dtx v1.0t		\array@parboxrestore: Added local settings of flags: dangerous! 734	
General: Move ltxcheck to separate file 14		\@iiminipage: Use it or lose it (@setminpage): Frank will want to lose it 736	
1996-09-28 ltmiscen.dtx v1.1f		1996-10-24 ltfloat.dtx v1.1p	
\@xobeysp: Moved to ltspace.dtx .. 686		\flo@floatboxreset: Added local settings of flags: dangerous! 823	
1996-09-28 ltspace.dtx v1.2t		\marginparreset: Added local settings of flags: dangerous! 829	
\@xobeysp: Moved from ltmiscen.dtx and redefined to use \nobreakspace 376		\xfloat: Added \nодокумент to trap floats in the preamble 820	
1996-09-29 ltfiles.dtx v1.1g		1996-10-24 ltoutput.dtx v1.1z	
\document: Added disabling of \nодокумент 384		\addtocurcol: Added \nobreak, etc as appropriate 1026, 1029	
1996-09-29 ltoutput.dtx v1.1x		\specialoutput: Added \nobreak as appropriate 1004	
\newpage: Checks for noskipsec and inlabel added 998		\topnewpage: Added \nодокумент to trap \twocolumn in the preamble 1000	
1996-09-29 ltsect.dtx 1.0w		\newpage: Better checks for noskipsec and inlabel added, plus nobreak . 998	
\@noskipsetrue: Added documentation 804		1996-10-25 llists.dtx v1.0n	
1996-09-30 ltoutput.dtx v1.1y		\endtrivlist: Change \indent to \leavevmode 718	
\newpage: Checks for noskipsec and inlabel removed pending further tests 998		Reset flags explicitly 718	
1996-10-04 ltclass.dtx v1.0v		1996-10-25 ltoutput.dtx v1.2a	
\RequirePackageWithOptions: Reset \@unprocessedoptions for /2269 895		\newpage: Reset all flags explicitly . 998	
1996-10-05 ltfiles.dtx v1.1h			
\clubpenalty: Added setting its value 382			
1996-10-08 lfntcmd.dtx v3.3u			
\DeclareTextFontCommand: Removed \check@icr when in vmode since it causes various errors (see pr/2157) 624			

1996-10-26 ltlists.dtx v1.0o		1996-11-18 ltoutenc.dtx v1.8d
\endtrivlist: Correct typo	718	General: (DPC) lowercase external file names. internal/1044
1996-10-27 ltoutenc.dtx v1.8c		446
\@strip@args: Removed macro	410	1996-11-20 fontdef.dtx v2.2p
General: Added \r A	420	General: lowercase fd and enc.def file names /1044
Added		598
\textasteriskcentered	416, 427	
Corrected syntax descriptions	404	1996-11-20 ltvers.dtx v1.0f
Removed \aa and \AA	416, 420, 423	General: Check for old format modified /2319
1996-10-28 lplain.dtx v1.1u		38
General: (CAR) More doc changes	15	1996-11-23 ltoutenc.dtx v1.8e
\dotfill: Removed math mode	33	General: Corrected description
1996-10-29 lplain.dtx v1.1v		404
\dotfill: Got arithmetic correct (CAR)	33	Extended description
1996-10-29 lspace.dtx v1.2u		405
\gnewline: Added macro	366	1996-11-28 ltvers.dtx v1.0g
\@no@lnbk: Macro replaces \@lnbk and \@nolnbk	364	General: Check for old format modified /2319
\@: Corrected and rationalised code	364	38
\nolinebreak: Reimplemented both using \@no@lnbk	363	1996-12-06 ltdirchk.dtx v1.0u
1996-10-31 ltfinal.dtx v1.0z		\IfFileExists: *** removed from various messages for GNU Make. internal/2338
General: Added extra \lcode, hoping it does no harm in T1 (pr/1969)	1066, 1073	10
1996-10-31 ltlists.dtx v1.0p		1996-12-06 ltfssini.dtx v3.0h
\@trivlist: Added check for missing item in outer list	717	General: (DPC) Remove *** from messages internal/2338
1996-10-31 ltsect.dtx v1.0y		596
General: Corrected and tidied documentation; removed long lines	803	1996-12-17 ltdefns.dtx v1.0w
1996-11-03 lplain.dtx v1.1w		\g@addto@macro: Use \begingroup to save making a mathord
\dotfill: Saved tokens by using \hb@xt@	33	112
1996-11-04 lterror.dtx v1.2m		1996-12-20 ltsect.dtx v1.0z
\@nodocument: Always define \@nodocument in kernel, so that it can be cleared by \document.	335	\@dottedtocline: Added \nobreak for latex/2343
1996-11-04 ltlists.dtx v1.0q		814
\@trivlist: Moved check for missing item: only checked when not inlabel flag is false	717	1997-01-08 fontdef.dtx v2.2q
1996-11-05 ltfiles.dtx v1.1i		General: Use \DeclareMathDelimiter to set delimiter codes
\nofiles: Standard \if@nobreak test added	387	607
1996-11-09 ltmath.dtx v1.1c		\mathparagraph: Define using \DeclareMathSymbol
\@ensuredmath: Made long, as it was before. /2104	705	616
1996-11-18 ltfsbsas.dtx v3.0s		1997-01-08 ltfiles.dtx v1.1j
\define@newfont: (DPC) lowercase fd file names. internal/1044	474	\@include: reset \deadcycles latex/2365
		391
		1997-01-08 ltmath.dtx v1.1d
		\root: (DPC) Remove spurious space tokens from plain TeX definition /2359
		695
		1997-02-05 ltdefns.dtx v1.0x
		\g@addto@macro: missing percent /2402
		112
		1997-02-21 ltlists.dtx v1.0r
		\@item: \ifvoid check added for \@noindent. latex/2414
		721
		1997-03-21 ltcounts.dtx v1.1e
		\fnsymbol: Use \mathsection and \mathparagraph. latex/2445
		454

1997-04-14 ltfiles.dtx v1.1k		1997-08-29 ltoutenc.dtx v1.9f
\document: Set the document space factor defaults. latex/2404	383	General: Added OT4 encoding, provided by Marcin Woliński.
\normalsfcodes: Macro added (from patch file) latex/2404	387	
1997-04-14 ltoutput.dtx v1.2b		1997-09-09 ltdefns.dtx v1.2z
\@writesetup: Call \normalsfcodes (from patch file) latex/2404	1013	\provide@command: Use \begingroup to avoid generating math ords if used in math mode. pr/2573
Move \label and \index (from patch file)	1013	1997-09-15 ltpictur.dtx v1.1g
1997-04-24 ltbibl.dtx v1.1m		\@getcirc: Warn if lines become invisible pr/2524
\@citex: \@empty to avoid primitive error on empty cite keys. latex/2432	844	\@picture@warn: Macro added pr/2524
1997-04-30 ltoutenc.dtx v1.9a		\@sline: Warn if lines become invisible pr/2524
General: Changed \textsc to \scshape	417	1997-10-06 lccounts.dtx v1.1f
Introduced \textcopyright and modified \copyright	417	\@Roman: Change \@Roman to be fully expandable, so that the result is written properly to files.
Introduced \textcopyright and modify \copyright	418	\@slowromancap: Macro added.
Modified \textunderscore, removing \mathunderscore	417	1997-10-08 ltlogos.dtx v1.1h
Modified \underscore, removing \mathunderscore	418	\LaTeX: Simplify macro (force loading of suitable math fonts once).
1997-04-30 ltoutenc.dtx v1.9b		1997-10-10 ltclass.dtx v1.0y
General: Added \leavevmode to \textunderscore	417	\endfilecontents: \@currenvir in banner
1997-05-04 ltoutenc.dtx v1.9c		\reserved@c not \verb@out to save a csname
General: Added ‘hex index tabs’	424	Check for text before or after \end environment. latex/2636
Added TS1 encoding v2.2.beta	430	Use \@gobbletwo
1997-05-07 ltoutenc.dtx v1.9d		1997-10-17 lfntcmd.dtx v3.3w
General: Added \leavevmode to \textcompwordmark	417	\check@nocorr@: Check for vertical mode moved here, from \DeclareTextFontCommand (see PR/2646).
1997-05-07 ltspace.dtx v1.2v		\DeclareTextFontCommand:
\newline: Made completely robust.	365	Reinstalled \check@icr as check is now done in \check@nocorr@ (see PR/2646).
1997-05-29 lfsstrc.dtx v3.0j		1997-10-20 ltfinal.dtx v1.1a
General: Replaced \\ by \MessageBreak, as suggested by Donald Arseneau.	510	\cuclclist: Removed \aa and \AA from \cuclclist as these are macros.
1997-05-29 ltlogos.dtx v1.1f		1997-10-21 ltdefns.dtx v1.2z1
\LaTeXe: Added \math so that the L ^A T _E X 2 _ε logo works with non-zero values of \mathsurround.	379	\renew@command: Use \begingroup/\endgroup rather than braces for grouping, to avoid generating empty math atom.
1997-06-16 ltdirchk.dtx v1.0v		1997-10-21 lfssbas.dtx v3.0t
General: documentation improvements mainly from internal/2520	1	\define@newfont: Move \makeatletter to \nfss@catcodes.
1997-06-16 ltfloat.dtx v1.1s		473
General: documentation fixes	816	
1997-06-16 lfntcmd.dtx v3.3v		
General: Fix typo in documentation.	622	
1997-08-05 ltoutenc.dtx v1.9e		
General: Corrected order of arguments in \UseTextSymbol example.	404	

\nfss@catcodes: Moved \makeatletter from \try@load@font@shape.	474	Removed default settings, see next section.	430
1997-11-09 ltoutput.dtx v1.2c \@specialoutput: Remove incorrect code: only one \emptycol is needed here	1002	1997-12-19 ltoutenc.dtx v1.9i General: Documentation corrections. 403	
\@topnewpage: Documentation of vsize check enhanced	999	1997-12-20 fontdef.dtx v2.2s General: Added documentation	600
1997-11-13 ltfsdcl.dtx v3.0f \DeclareSymbolFont: (DPC) Really update \group@list don't leave new version in \toks@. latex/2661 551		1997-12-31 ltoutenc.dtx v1.9k General: Further correction	404
\stepcounter: (DPC) Remove as never used. (Re)defined in ltcounts	540	1998-01-12 ltoutenc.dtx v1.9k General: Added \ProvidesPackage for textcomp.sty	403
1997-11-19 ltfloat.dtx v1.1t \@footnotetext: Missing percent, again	835	Adding missing braces and \ushape.	432
1997-11-19 ltoutput.dtx v1.2d \@vtryfc: Reindent code, to be understandable(DPC).	1019	1998-01-16 ltoutenc.dtx v1.9m General: fixed decimal codes. latex/2734	428
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)	543	1998-03-04 ltdefns.dtx v1.2z2 \@xargdef: Unnecessary \expandafter removed: pr/2758 . 84	
\select@group: (DPC) inline use of \stepcounter (faster, and saves a csname per math version as no reset list)	541	1998-03-05 ltoutenc.dtx v1.9n General: Added masc/fem ords as in pr/2579	417
1997-11-23 ltoutenc.dtx v1.9g General: Use \textperthousand, \textpertenthousand and \textfractionsolidus not \textpermill, \textperenmill and \textfraction. /2673	430	1998-03-20 ltdefns.dtx v1.2z3 \@thirdofthree: Macro added	89
1997-12-17 ltoutenc.dtx v1.9h General: Added \textperthousand and \textpertenthousand . 421, 422		1998-03-20 ltoutenc.dtx v1.9o General: Documentation added about order of decls	406
Added code for textcomp.sty.	446	Documentation added for pr/2783 405	
Added section.	446	\UndeclareTextCommand: Macro added for pr/2783	414
Added textcomp.sty.	403	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.	422	\UndeclareTextCommand declarations for pr/2783	660
Changed to decimal codes in \oalign.	432	Load decls after defaults for speed. 660	
Changed to decimal codes.	428	1998-03-21 ltclass.dtx v1.0z General: Added to documentation of	
Documentation changes and additions.	403	filecontents	874
Example corrected, braces removed.	403	1998-03-21 ltclass.dtx v1.1a \@providesfile: Allow & Internal/2702	888
		General: Correct to new onlypreamble command list	923
		1998-03-25 ltfsbas.dtx v3.0u \showhyphens: Suppress unnecessary error when used in preamble	483
		1998-04-11 fontdef.dtx v2.2t General: Added \mathring accent (pr2785)	614
		1998-04-15 fontdef.dtx v2.2u General: Use new syntax for \DeclareMathDelimiter	607

1998-04-15 ltfssdcl.dtx v3.0h		1998-08-17 ltdirchk.dtx v1.0w
\@xxDeclareMathDelimiter: Macro added (pr/2662)	562	General: (RmS) Documentation improvements.
1998-04-17 fontdef.dtx v2.2v		1998-08-17 lfntcmd.dtx v3.3x
General: Reinsert symbol defs for < and > chars.	608	General: (RmS) Minor documentation fixes.
1998-04-18 fontdef.dtx v2.2w		1998-08-17 lfssbas.dtx v3.0v
General: Reinsert symbol def for / char.	608	General: (RmS) Documentation fixes. .
1998-05-07 ltclass.dtx v1.1b		1998-08-17 lfssdcl.dtx v3.0i
\@onefilewithoptions@clashchk: Modify help message for latex/2805	901	General: (RmS) Corrected minor glitches in changes entries.
1998-05-18 ltab.dtx v1.1j		1998-08-17 lfssini.dtx v3.0i
\@endpbox: Use \setlength to set \hsize, so that the changes in the calc package apply here.	765	General: (RmS) Minor documentation fixes.
\@tabular*: Use \setlength, so that calc extensions apply.	754	1998-08-17 ltlogos.dtx v1.1i
1998-05-20 ltfinal.dtx v1.1b		General: (RmS) Minor documentation fixes.
General: Set up lccodes before loading hyphenation files: pr/2639	1065	1998-08-17 ltmath.dtx v1.1c
Set up uc/lccodes after loading hyphenation files: pr/2639	1073	General: (RmS) Minor documentation fixes.
1998-05-28 lterror.dtx v1.2n		1998-08-17 ltmiscen.dtx v1.1g
\@notdefinable: Added message re ‘end..’ pr/1555	335	General: (RmS) Minor documentation fixes.
1998-06-04 ltboxes.dtx v1.1c		1998-08-17 ltspace.dtx v1.2w
\@rule: Support calc-expressions . .	739	General: Documentation fixes.
1998-06-12 ltoutenc.dtx v1.9p		1998-08-17 preload.dtx v2.1g
General: Corrected 130 and 131, see pr/2834	433	General: (RmS) Minor documentation fixes.
Renamed \textmacron pr/2840 . .	434	1998-09-19 ltoutenc.dtx v1.9r
1998-06-12 ltoutenc.dtx v1.9q		\@a: Added \string (pr/2878)
\add@accent: Explicitly set \spacefactor after \accent (pr/2877)	410	1998-11-13 ltab.dtx v1.1m
1998-06-12 lttextcomp.dtx v1.9p		\@array: Check for hmode to see if something went wrong during parsing (pr/2884)
General: Renamed \textmacron pr/2840	656	1999-01-05 fontdef.dtx v2.2x
1998-06-18 ltab.dtx v1.1k		General: Need special protection for character > in \changes entry.
General: Small addition to documentation	742	1999-01-06 lfssbas.dtx v3.0w
1998-07-06 ltab.dtx v1.1l		\@DeclarFontEncoding: Added \LastDeclaredEncoding to support cyrillic integration (pr/2988)
General: Small correction to documentation	742	\@LastDeclaredEncoding: Added \LastDeclaredEncoding to support cyrillic integration (pr/2988)
1998-08-17 ltboxes.dtx v1.1e		1999-01-06 ltoutenc.dtx v1.9r
General: (RmS) Minor Documentation fixes.	725	\@strip@args: New impl for latex/2930
1998-08-17 ltclass.dtx v1.1c		General: Minor documentation fix.
General: (RmS) Minor documentation fixes.	874	1999-01-06 ltoutput.dtx v1.2e
		\@makecol: Added negative vskip, as when processing outputbox below:

suggested by Fred Bartlett pr/2892	1008	New macro added	85
1999-01-07 ltdefns.dtx v1.3a \@ifnextchar: made long	107	1999-06-10 ltoutenc.dtx v1.9u General: Ensure that we also forget old options (pr/2888)	448
\@newenvb: made long and brace optional arg. latex/2896	87	1999-06-12 ltoutenc.dtx v1.9v General: Extend \@uclclist only once	447
\@testopt: made long and brace optional arg. latex/2896	84	1999-10-09 ltmath.dtx v1.1e \active@math@\prime: Macro added, see PR 3104.	700
1999-01-07 ltdefns.dtx v1.3b \@ifnextchar: extra \long. latex/2902	107	\prime@s: Introduce \active@math@\prime.	700
1999-01-07 ltoutenc.dtx v1.9r General: Hackery to allow using fontenc several times	448	1999-10-09 ltoutput.dtx 1.2f \@activechar@info: Reset definition of active prime character (used in math mode)	1011
Hackery to temp support cyrillic uc/lc	446	1999-10-28 ltoutenc.dtx v1.9w \add@accent: Give \accent@spacefactor a default definition (pr/3084)	410
1999-01-13 ltoutenc.dtx v1.9s \@strip@args: Simplified solution for latex/2930	412	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.	446
1999-01-18 ltdefns.dtx v1.3c \@yargd@f: New implementation DPC /2942	85	2000-01-07 ltmiscen.dtx v1.1h \@verbatim: Disable hyphenation even if the font allows it.	687
1999-02-09 ltdefns.dtx v1.3d \@yargd@f: catch bad argument forms by re-inserting #3	85	2000-01-15 ltpictur.dtx v1.1i \@upvector: Removed space at end-of-line, CAR	780
1999-02-12 lttextcomp.dtx v3.0j \legacyoldstylenums: Use \rmdefault instead of cmm (pr/2954)	631	2000-01-30 ltfntcmd.dtx v3.3y \DeclareTextFontCommand: Use \hmode@bgroup now (pr/3160) . . .	624
1999-02-24 ltoutenc.dtx v1.9t General: Corrected hackery cyrillic uc/lc list	446	2000-01-30 ltoutenc.dtx v1.9y General: Use \hmode@bgroup where applicable (pr/3160)	419–422, 427–430, 432
1999-03-01 ltdefns.dtx v1.3e \@ifnextchar: remove extra \long. internal/2967	107	\add@accent: Use \hmode@bgroup where applicable (pr/3160)	409
1999-04-15 ltpictur.dtx v1.1h \@getlarrow: Replaced octal number, CAR	779	\hmode@bgroup: Macro added	410
\@upvector: Replaced octal number, CAR	780	2000-01-30 ltoutenc.dtx v1.9z \@use@text@encoding: Macro reimplemented (pr/3160)	412, 413
General: Replaced octal number, CAR	779, 780	\add@accent: Macro reimplemented (pr/3160)	409
Replaced octal numbers, CAR	767	\hmode@start@before@group: Macro added (pr/3160)	413
1999-04-19 ltfloat.dtx v1.1u \caption: Made caption an error outside a float: latex/2815	819	2000-05-19 ltmiscen.dtx v1.1i \enddocument: Reset \AtEndDocument for latex/3060	672
1999-04-27 ltboxes.dtx v1.1f \parboxto: (CAR) Changed \empty to \relax as flag for natural width: pr/2975	734		
1999-04-29 ltdefns.dtx v1.3f \@yargd@f: Full expansion and conversion needed for digit in new version, see pr/3013	85		

2000-05-26 ltpage.dtx v1.0j		2001-02-16 ltxref.dtx v1.1k	
\@markright: Reimplementation to fix expansion error (pr/3203).	872	\@newl@bel: Added an extra grouplevel (PR3250), jlb	667
\leftmark: Use \empty instead of brace group (pr/3203).	872	2001-05-25 ltclass.dtx v1.1d	
\markright: Reimplementation to fix expansion error (pr/3203).	870	\@providesfile: Explicitly set catcode of \endlinechar to 10 (pr/3334)	888
\rightmark: Use \empty instead of brace group (pr/3203).	872	2001-05-25 ltdirchk.dtx v1.0x	
2000-06-02 ltpage.dtx v1.0k		General: Explicitly set catcode of \endlinechar to 10 (pr/3334) . . .	4
\@markright: Small adjustment to give slightly less expansion, CAR . . .	872	2001-05-28 ltoutenc.dtx v1.93	
\markright: Small adjustment to give slightly less expansion, CAR	870	General: Added composites for compatibility with T1, pr/3295 . . .	421
Tidied 1.0j reimplementation, CAR . . .	870	Changed the effect of \.\i, pr/3295	424
2000-07-11 ltmiscen.dtx v1.1j		2001-06-02 fontdef.dtx v2.2y	
\@enddocument@kernel@warnings: Fix typo in warning	674	General: Provide default cfg files (pr/3264)	617
2000-07-12 ltoutput.dtx 1.2g		2001-06-04 fontdef.dtx v2.2z	
General: Ensure that rule is in \normalcolor	1052	General: Guard against math active equal and pipe sign in \models (pr/3333)	613
2000-07-12 ltoutput.dtx 1.2i		Guard against math active equal sign in \Relbar (pr/3333)	613
\@makecol: Removed negative vskip, as it gives unacceptable results when the depth is large: pr/3189 . . .	1008	2001-06-04 ltclass.dtx v1.1e	
2000-07-19 ltoutput.dtx v1.2h		\@providesfile: But only if it is a char (pr/3334)	888
\@writesetup: Reset and restore \@if@newlist for internal/3231 . . .	1012	2001-06-04 ltdirchk.dtx v1.0y	
2000-08-23 ltfinal.dtx v1.1c		General: But only if it is a char (pr/3334)	4
General: Fix typo in warning	1067	2001-06-04 ltpictur.dtx v1.1j	
2000-08-30 ltoutenc.dtx v1.91		\@sline: Don't warn for exactly zero pr/3318	777
\@use@text@encoding: Rearranged but no change to final code, CAR (pr/3160)	412	2001-06-04 ltvers.dtx v1.0i	
\add@accent: Rearranged but no change to final code, CAR (pr/3160)	409	General: Check for old format disabled	38
2000-09-01 ltfinal.dtx v1.1d		2001-06-05 ltoutenc.dtx v1.94	
\errhelp: Set error help empty at very end (pr/449 done correctly).	1077	General: Text composite Commands need kludges for ',' – see tbl1903.lvt	421
2000-09-24 ltfloat.dtx v1.2b		2001-08-26 ltclass.dtx v1.1f	
\end@dblfloat: FMi: use output routine to defer float	824	\@providesfile: Readded setting of space char (pr/3353)	888
2000-09-24 ltoutput.dtx v1.2b		2002-02-24 lplain.dtx v1.1x	
\@doclearpage: FMi: ensure \doclearpage is called again until all floats are output.	1006	\loggingall: Macro added	34
2000-09-24 ltoutput.dtx v1.2n		\loggingoutput: Macro added	34
\@addtocurcol: FMi: test for wide float was in wrong place	1025	\showoutput: Use newly added \loggingoutput	34
2001-01-07 ltoutput.dtx v1.2j		\tracingall: Use newly added \loggingoutput	34
\@writesetup: And do it in the right macro (pr/3286)	1012	2002-06-16 ltoutenc.dtx v1.95	
		General: Added \textbardbl (pr/3400)	427
		Added default for \textbardbl (pr/3400)	416

2002-06-17 ltoutenc.dtx v1.95		2004-01-03 ltoutenc.dtx v1.99b	
General: Corrected \c for T1 (pr/3442)	422	General: Added \textogonekcentered (pr/3532)	422
Definition of \texttexclamdown changed (pr/3368)	420	Added composites for \k (pr/3532) 427	
Definition of \textquestiondown changed (pr/3368)	420	Use \oalign for \k (pr/3532) ... 422	
2002-06-18 ltoutenc.dtx v1.95		2004-01-04 ltbibl.dtx v1.1p	
General: Changed def for \textregistered to avoid small caps (pr/3420)	417	\nocite: Changed error message ... 845	
2002-10-01 lftfloat.dtx v1.1v		2004-01-04 ltoutenc.dtx v1.99c	
\thempfootnote: Use braces around \itshape to keep font change local (pr/3460).	833	General: More adjustments for ogonek (pr/3532)	422
2002-10-02 ltfssbas.dtx v3.0x		2004-01-23 ltdefns.dtx v1.1g	
\DeclareFontSubstitution: Adding \LastDeclaredEncoding introduced a bug as on some occasions that macro name was stored in the internal lists instead of the actual encoding. (pr/3459) 463		\newenva: Use kernel version of \ifnextchar (pr/3501)	87
2002-10-28 ltlists.dtx v1.0s		\testopt: Use kernel version of \ifnextchar (pr/3501)	84
\endtrivlist: Check for math mode (pr/3437)	718	\xargdef: Use kernel version of \ifnextchar (pr/3501)	84
2002-10-28 ltoutenc.dtx v1.96		\xdblarg: Use kernel version of \ifnextchar (pr/3501)	108
General: coding change, to follow bug fix by DEK in plain.tex (pr/3469)	420, 429	2004-01-23 ltdefns.dtx v1.3g	
2002-12-13 ltbibl.dtx v1.1n		\kernel@ifnextchar: Added macro (pr/3501)	107
\citex: Added \leavevmode in case citation is at start of paragraph (pr/3486)	844	2004-01-28 ltclass.dtx v1.1g	
2003-01-01 ltfntcmd.dtx v3.3z		\providesfile: Use kernel version of \ifnextchar (pr/3501)	888
General: Code checked and documentation extended by Chris 624		2004-01-28 ltvers.dtx v1.0k	
2003-05-18 ltbibl.dtx v1.1o		General: Check for old format made 5 years (pr/3601)	38
\nocite: Check if we are after \document	845	2004-02-02 fontdef.dtx v2.3	
2003-08-27 ltpictur.dtx v1.1k		General: Many things from here on made robust	612
\bezier: added missing displacement pr/3566	796	2004-02-02 ltoutenc.dtx v1.99	
\sline: check for \linechar being empty pr/3570	777	General: Added \textbigcircle ... 427	
2003-10-13 ltfinal.dtx v1.1e		2004-02-04 fontdef.dtx v2.3a	
General: Added extra \lccode for \-		General: Added bigtriangle synonyms for stmaryrd	610
and \textcompwordmark	1066	2004-02-04 ltspace.dtx v1.3	
2003-12-16 ltoutput.dtx v1.2k		\nobreakdashes: (Macro added ... 375	
\makecol: Ensure that \elt has a defined state (pr/3586)	1008	2004-02-06 ltoutenc.dtx v1.99d	
2003-12-30 ltpictur.dtx v1.1j		\cinmathwarn: New command added to fix severe bug: pr/3563 ... 407	
\getcirc: issue warning if circle size can't be met pr/3473	788	2004-02-07 ltoutput.dtx v1.2l	
		\docclearpage: Empty klugeins box if necessary, pr/3528	1006
		2004-02-13 ltoutenc.dtx v1.99e	
		General: Documentation fixes: typos 403	
		2004-02-15 ltbibl.dtx v1.1q	
		\cite@ofmt: Added hook with default value \hbox	846
		\citex: Changed to use a hook with default value \hbox	844
		2004-02-15 ltspace.dtx v1.3a	
		\nobreakdashes: (Added spacefactor setting	375

2004-10-20 ltoutput.dtx v1.2m	\@makecol: Removed dead code . . .	1008	2009-12-14 ltfnntcmd.dtx v3.4a	\ifmaybe@ic: Macro added	627
2005-07-27 ltfssdcl.dtx v3.0j	\DeclareMathAlphabet: (MH) Make document commands robust . . .	554	\maybe@ic@: Use switch \ifmaybe@ic instead of \if@tempswa	627	
	\DeclareSymbolFontAlphabet: (MH) Make document commands robust . . .	567	\t@st@ic: Use switch \ifmaybe@ic instead of \if@tempswa	628	
	\new@mathalphabet: (MH) Make document commands robust . . .	555			
	\non@alphe@r: (MH) Change because command is now properly robust . . .	542			
	\SetMathAlphabet: (MH) Make document commands robust . . .	556			
2005-09-27 ltoutenc.dtx v1.99g	General: Replace \sh@ft by \ltx@sh@ft	419, 422, 429			
2005-09-27 ltplain.dtx v1.1y	\ltx@sh@ft: New macro	33			
	\sh@ft: Macro no longer used but left for compatibility	33			
2005-11-08 ltoutenc.dtx v1.99h	General: Added \ij and \IJ from babel. (pr/3771)	416, 421, 423			
2005-11-10 ltmath.dtx v1.1g	\L: (MH) Fixed potential problem in \L (pr/3399).	701			
	General: (MH) Minor documentation fixes.	693			
2006-05-18 ltboxes.dtx v1.1g	\@parboxto: Ensure \@parboxto holds the value of \tempdimb not the register itself (pr/3867)	734			
2006-09-13 ltoutput.dtx v1.1m	General: Ensure that rule is in \normalcolor	1053			
2007-08-05 ltclass.dtx v1.1h	\@fileswithoptions: Prevent loss of brackets PR/3965	897			
2007-08-06 ltcntrl.dtx v1.0h	\@fornoop: Really make defs long . .	327	\f@tracemessage: Renamed internal trace commands; provide as package	1043	
2007-08-31 ltfssdcl.dtx v3.0l	\SetSymbolFont@: Font warning changed to info for encoding change (pr/3975)	553	\end@dblf@t: Inline the code to allow some coexistence with packages that hook into \end@float and do not know about the algorithm change	824	
2009-09-24 ltvers.dtx v1.0l	General: Stop checking for old format . .	38	\end@dblf@t: missing \fi added . .	824	
2009-10-20 ltfssdcl.dtx v3.0m	\in@: More robust thanks to Heiko.	538	\newmarks: macro added	1060	
2009-10-28 lttextcomp.dtx v1.99k	General: Added Latin Modern and TeX Gyre subsets	662	\newXeTeXintercharclass: macro added	1061	
2009-11-04 lttextcomp.dtx v1.99l	General: Added more Latin Modern and TeX Gyre subsets	662	\textsubscript: Command added (latexrelease)	835	
			\textsubscript: Command added (latexrelease)	834	
			\mathgroup: move allocation to lplain.	459	

2014-12-30 ltoutput.dtx v1.2m		2015-01-10 lccounts.dtx v1.1h	
General: Command updated (latexrelease)	1052	\@fnnsymbol: Unse \TextOrMath (latexrelease)	455
2014-12-30 lplain.dtx v2.0a		\@stpelt: Reset all within counters in one go (latexrelease)	451
\@alloc: macro added	20	2015-01-11 lccounts.dtx v1.1h	
\@alloc@chardef: macro added	19	\TextOrMath: Add command to solve robustness issues (pr/3752) (latexrelease)	456
\@alloc@top: macro added	19	2015-01-11 lffloat.dtx v1.2b	
\@ch@ck: macro added	20	\@dblfloatplacement: float order in 2-column (latexrelease)	826
\@extrafloats: macro added	21	\@xfloat: Check for valid option (latexrelease)	820
\newlanguage: New engine-specific allocation scheme (latexrelease)	18	\end@dblfloat: float order in 2-column (latexrelease)	824
2014-12-30 ltspace.dtx v1.3b		2015-01-11 lfssbas.dtx v3.0y	
\@: \@ discards spaces when moving (pr3039)(latexrelease)	376	\@DeclarMathSizes: Allow arbitrary units (latexrelease)	466
2015-01-03 ltdirchk.dtx v1.4a		2015-01-11 ltspace.dtx v1.3d	
\typein: use modified definition in luatex	82	\@Ephack: Allow hyphenation (Donald Arseneau pr/3498) (latexrelease)	369
2015-01-03 ltdirchk.dtx v1.1		\@esphack: Allow hyphenation (Donald Arseneau pr/3498) (latexrelease)	368
General: Enable extra primitives when LuaTeX is used	3	2015-01-14 ltoutput.dtx v1.2n	
2015-01-03 ltfinal.dtx v2.0a		\@addtocurcol: float order in 2-column (latexrelease)	1024
General: Skip resetting codes with Unicode engines	1073	\@addtobdblcol: float order in 2-column (latexrelease)	1034
Unicode data loading added	1064	\@adtonextcol: float order in 2-column (latexrelease)	1031
2015-01-07 ltvers.dtx v1.0n		\@doclearpage: Empty kludgeins box if necessary, pr/3528	1005
\@check@IncludeInRelease: macro added	40	\@startdblcolumn: float order in 2-column (latexrelease)	1018
2015-01-08 ltboxes.dtx v1.1h		\@xtryfc: float order in 2-column (latexrelease)	1020
\framebox: Make Robust (latexrelease)	732	\@ztryfc: float order in 2-column (latexrelease)	1021
\makebox: Make Robust (latexrelease)	726	2015-01-14 ltspace.dtx v1.3e	
\parbox: Make Robust (latexrelease)	733	\@addpenalty: Avoid adding redundant skips (DPC)	372
\raisebox: Make Robust (latexrelease)	739	2015-01-17 ltvers.dtx v1.0m	
\rule: Make Robust (latexrelease)	738	\@check@IncludeInRelease: modified with \@currname	40
\savebox: Make Robust (latexrelease)	729	2015-01-19 ltvers.dtx v1.0o	
2015-01-08 ltdfns.dtx v1.4a		\@check@IncludeInRelease: Optional argument	40
\MakeRobust: Added macro	92		
2015-01-08 ltlenth.dtx v1.1c			
\setlength: to ensure first length argument is terminated. (latexrelease)	457		
2015-01-08 ltmath.dtx v1.1h			
\@: Make Robust (latexrelease)	701		
\]: Make Robust (latexrelease)	701		
2015-01-09 lfssini.dtx v3.1a			
\em: Allow \emph to produce small caps (latexrelease)	590		
\emminershape: macro added (latexrelease)	590		
2015-01-09 ltspace.dtx v1.1h			
\addpenalty: Donald Arseneau's fix from PR/377703 (latexrelease)	372		

2015-01-20 ltoutput.dtx v1.2m \f@tracemessage: Reset \IncludeInRelease flags 1044	2015-02-21 ltpictur.dtx v1.1k General: Removed autoload code 767
2015-01-22 ltvers.dtx v1.0p General: Preserve any \everyjob material inserted by a loader (.ini file) 39	2015-02-21 lplain.dtx v2.0e General: Removed autoload code 15
2015-01-23 ltfinal.dtx v2.0b \newmarks: use reserved count 256 1060 \newXeTeXintercharclass: use reserved count 257 1061	2015-02-21 ltab.dtx v1.1n General: Removed autoload code 742
2015-01-23 lplain.dtx v2.0c \xtrafloats: reserve counts 256–265 21	2015-02-21 ltvers.dtx v1.0r General: Removed autoload code 38
2015-01-24 ltfinal.dtx v2.0c General: Skip T1-code entirely with Unicode engines 1064	2015-02-21 ltvers.dtx v1.0w \@check@IncludeInRelease: set \@currname empty here (in case \IncludeInRelease input early) 40
2015-02-03 ltfinal.dtx v2.0d General: Set \lccode for - with Unicode engines 1065	2015-02-22 ltfsscmp.dtx v3.0e General: Moved all code into latexrelease - obsolete commands are no longer automatically part of the kernel 533
2015-02-16 ltoutenc.dtx v1.99m General: Added \textcommabelow latex/4414 418	2015-03-02 lplain.dtx v2.0f \@mathgroup@top: macro added 20 \newlanguage: allow 255 math groups in Unicode engines 18
2015-02-16 ltoutenc.dtx v1.99n General: Added \textcommaabove 419 Added composites for ç 427 Added composites for \c 421	2015-03-10 lplain.dtx v2.0g \hideoutput: macro added 37 \loggingall: Reorganize to be less noisy 34 \tracingnone: macro added 35
2015-02-16 lttextcomp.dtx v1.99m General: Added lmtt (Heiko Oberdiek) latex/4415 662	2015-03-12 ltoutput.dtx v1.2m General: initialise \@dbldeflist again 995
2015-02-19 ltvers.dtx v1.0q \@check@IncludeInRelease: Swap argument order 40	2015-03-18 ltfssdcl.dtx v3.0q \DeclareSymbolFont: Restrict Symbol fonts to 0-15 551
2015-02-20 lplain.dtx v2.0d \loggingall: Spell commands correctly :-) 34	\document@select@group: Introduce \@mathgroup@top 543
2015-02-21 ltdefns.dtx v1.4b General: Removed autoload support 80	\select@group: Introduce \@mathgroup@top 541
2015-02-21 lterror.dtx v1.2o General: Removed autoload support 328	2015-03-26 ltfinal.dtx v2.0d General: Use renamed unicode-letters.def 1064
2015-02-21 ltfiles.dtx v1.1m General: Removed autoload support 380	2015-04-07 ltfssbas.dtx v3.1a \wrong@fontshape: Try loading fd file if family has changed 478
2015-02-21 ltfssbas.dtx v3.0z General: Removed autoload code 459	2015-04-28 ltfinal.dtx v2.0f \newXeTeXintercharclass: define \@alloc@intercharclass for compatibility with older xelatex initialization 1061
2015-02-21 ltfsscmp.dtx v3.0d General: Removed autoload code 533	2015-05-10 ltlists.dtx v1.0t \@doendpe: Explicitly reset \clubpenalty before clearing \everypar; see also pr/0462 and pr/4065 719
2015-02-21 ltfssdcl.dtx v3.0p General: Removed autoload code 538	
2015-02-21 lfsstrc.dtx v3.0k General: Removed autoload code 508	
2015-02-21 ltoutenc.dtx v1.99m General: Removed autoload code 403	
2015-02-21 ltoutput.dtx v1.2n General: Removed autoload code 984	
\f@depth: macro added(latexrelease) 1004	

2015-06-19 ltfinal.dtx v2.0g	\newluafunction: Macro added	52
\@alloc@intercharclass@top: Use –1 for first range to get contiguous allocation	1061	
\newmarks: Use –1 for first range to get contiguous allocation	1060	
2015-06-19 ltplain.dtx v2.0h		
General: delete spurious old definition of \newtoks	24	
\@alloc: extra braces in case arguments not single token	20	
\newlanguage: Use –1 for first range to get contiguous allocation	18	
2015-06-23 ltfinal.dtx v2.0h		
General: set \patch@level in ltvers rather than in ltfinal/ltpatch	1076	
2015-06-23 ltvers.dtx v1.0t		
General: set \patch@level in ltvers rather than in ltfinal/ltpatch	38	
2015-08-06 ltplain.dtx v2.0i		
\xtraffloats: Add \string in case argument is not an unexpandable primitive	21	
2015-08-23 ltdirchk.dtx v1.2		
General: Do not use luatex prefix	3	
2015-08-23 ltvers.dtx v1.0v		
General: Allow negative patchlevel for pre-release	39	
2015-08-30 ltplain.dtx v2.1a		
\newinsert: new \newinsert implementation	23	
2015-09-205 ltoutput.dtx v1.3a		
General: extended \freelist	994	
2015-09-24 ltluatex.dtx v1.0a		
call_callback: Function added	68	
callback.register: Function modified	65	
callback_descriptions: Function added	72	
\catcodetable@atletter: Macro added	50	
\catcodetable@initex: Macro added	50	
\catcodetable@latex: Macro added .	50	
\catcodetable@string: Macro added	50	
add_to_callback: Function added	69	
remove_from_callback: Function added	71	
new_attribute: Function added	58	
disable_callback: Function added	72	
in_callback: Function added	72	
\newattribute: Macro added	50	
\newcatcodetable: Macro added	50	
\newluabytecode: Macro added	53	
\newluachunkname: Macro added	53	
2015-10-02 ltdirchk.dtx v1.2a	\newwhatsit: Macro added	52
General: Allow backing out of unprefixed names	3	
2015-10-02 ltluatex.dtx v1.0b		
General: Fix backing out of TeX code	54	
2015-10-02 ltluatex.dtx v1.0c		
General: Allow backing out of Lua code	54	
2015-10-02 ltluatex.dtx v1.0e		
uninstall: Function added	73	
2015-10-03 ltluatex.dtx v1.0f		
provides_module: use luatexbase_log	55	
2015-10-27 ltplain.dtx v2.1b		
\xtraffloats: Use global assignment when switching to extended range	21	
2015-11-07 ltspace.dtx v1.3f		
\esphack: Only space if there is no space at the end of the hlist latex/4443	368	
2015-11-14 ltluatex.dtx v1.0g		
General: Track LuaTeX changes for (token).create	57	
2015-11-18 ltplain.dtx v2.2a		
\newlanguage: Extended stream allocation in luatex (0.85)	18	
2015-11-19 ltplain.dtx v2.2b		
\newlanguage: Only extend allocation of write streams (see luatex list)	18	
2015-11-27 ltluatex.dtx v1.0h		
callback_descriptions: Match test in in-callback latex/4445	72	
in_callback: Guard against undefined list latex/4445	72	
2015-11-29 ltluatex.dtx v1.0i		
General: Declare this as local before used in the module error definitions (PHG)	55	
call_callback: Check name is not nil in error message (PHG)	68	
create_callback: Check name is not nil in error message (PHG)	68	
2015-12-02 ltluatex.dtx v1.0j		
General: Adjust hashtokens to store the result of tex.hashtokens(), not the function (PHG)	57	

Assorted typos fixed (PHG)	47	2016-02-18 ltfssdcl.dtx v3.0r
Declaration/use of first_ head fixed (PHG)	56	\@DeclareMathDelimiter: Check for delimiter not \delimiter 562
Remove nonlocal iteration variables (PHG)	47	\@DeclareMathAccent: Check for mathaccent not \mathaccemt 558
Remove unreachable code after calls to error() (PHG)	47	\@DeclareMathRadical: Check for radical not \radical 565
2015-12-02 ltluatex.dtx v1.0k		\@DeclareMathSymbol: Check for mathchar not \mathchar 560
General: resolve name and i.description (PHG)	66	2016-03-13 ltluatex.dtx v1.0n
call_callback: Give more specific error messages (PHG)	68	General: contribute_filter added 64
add_to_callback: Give more specific error messages (PHG)	69	insert_local_par added 64
remove_from_callback: adjust initialization of cb local (PHG)	71	2016-03-29 ltpictur.dtx v1.11
Give more specific error messages (PHG)	71	\@oval: add setting of line tests 789, 790
create_callback: Give more specific error messages (PHG)	68	initialise tests 789
2015-12-10 ltfinal.dtx v2.0i		\@ovhorz: use glue not leaders if horizontal line not required 792
General: Use new common Unicode data loaders	1064	\@ovvert: use glue not leaders if vertical line not required 791
2015-12-18 ltluatex.dtx v1.0l		\if@ovhline: macro added (latex/4452) 789
General: Load Unicode data from source	50	\if@ovvline: macro added (latex/4452) 789
2016-01-04 ltfinal.dtx v2.0j		2016-04-22 ltfinal.dtx v2.0q
General: Do not set up inter character classes for XeTeX	1064	\e@alloc@intercharclass@top: XeTeX 0.99996 has 4096 char classes not 256 1061
\@alloc@intercharclass@top: Start allocation at one not three	1061	2016-06-19 ltoutenc.dtx v1.99m
2016-01-05 ltfinal.dtx v2.0k		General: OT1 definition (was duplicate T1 definition) 421
\@alloc@intercharclass@top:		2016-06-20 ltclass.dtx v1.1j
Remove duplicated code	1061	General: don't declare as \onlypreamble 884
2016-01-05 ltfinal.dtx v2.0l		2016-07-29 lplain.dtx v2.2c
General: Correct latexrelease guards	1064	\extrafloats: use \global \chardef 21
Ensure old definitions for inter-character class toks are available using latexrelease	1064	\newinsert: fix for tlb-newinsert-001 23
Missing brace	1064	2016-10-02 ltclass.dtx v1.2a
2016-01-05 ltfinal.dtx v2.0m		\@ifclasswith: Ignore spaces while checking for option clash 885
General: Undefine XeTeX classes when using patching an older kernel	1064	\ExecuteOptions: Ignore spaces in argument 893
2016-01-05 ltfinal.dtx v2.0p		2016-10-15 ldirchk.dtx v1.2b
General: Only apply XeTeX change if XeTeX is in use	1064	General: Require eT _{EX} 4
2016-02-11 ltluatex.dtx v1.0m		2016-10-15 lterror.dtx v1.2p
General: pdf_stream_filter_callback removed	64	General: Require eT _{EX} 328
process_rule, [hv]pack_quality append_to_vlist_filter added	64	2016-10-15 ltfinal.dtx v2.0r
read_cidmap_file added	63	General: Require eT _{EX} 1060
show_warning_message added	64	2016-10-15 lplain.dtx v2.2d
token_filter removed	63	General: Require eT _{EX} 15

2016-10-16 lplain.dtx v2.3a	\newlanguage: Allow languages up to 16383 in luatex	18	2017-02-19 ltoutenc.dtx v2.0f	General: add \empty to guard against 3rd argument being empty	420		
2016-10-19 lcounts.dtx v1.1j	\TextOrMath: Test directly for \protected	456	declare composites with empty base for hat and tilde, use same slots for \textasciicircum ans \textasciitilde	435	declare straight quotes using new \remove@tlig command	435	
2016-11-06 lplain.dtx v2.3b	General: Drop \outer entirely	15	2017-02-22 ltoutenc.dtx v2.0g	General: Fix typo introduced at 2.0f	435		
2016-11-09 ltclass.dtx v2.1b	\@fileswithoptions: Improve \ifx tests PR/4497	897	2017-02-24 ltoutenc.dtx v2.0h	General: introduce \DeclareUnicodeAccent	435		
2016-11-17 ltluatex.dtx v1.0p	General: call_edit added	64	\DeclareTextCompositeCommand: add check whether the accent command is defined for this encoding	410	2017-03-08 ltclass.dtx v1.2c	General: add \@parse@version@dash to support yyyy-mm-dd as well as yyyy/mm/dd	884
2016-12-03 fontdef.dtx v3.0a	General: (DPC) Default to TU encoding for Unicode TeX engines	600	2017-03-09 ltfinal.dtx v2.0t	\l@nohyphenation: ensure \l@nohyphenation is defined. . .	1067		
2016-12-04 ltoutenc.dtx v2.0a	\shapedefault: (DPC) Default to TU encoding for Unicode TeX engines	604	2017-03-09 ltmiscen.dtx v1.1m	\overline: Use \language not \hyphenchar	687		
2017-01-01 ltoutput.dtx v1.3b	General: Added TU encoding	435	\verb: Use \language to stop hyphenation	691	2017-03-10 ltfiles.dtx v1.1n	\document: Save language default . .	383
2017-01-10 ltfsbas.dtx v3.2a	General: make fpmin negative so ignored even if float height is negative	1052	2017-03-10 ltoutput.dtx v1.3c	\@writesetup: Reset \language . .	1012		
2017-01-23 ltoutenc.dtx v2.0b	\showhyphens: Add version of \showhyphens that works with XeTeX	483	2017-03-13 ltdefns.dtx v1.5a	\-: Define \- in terms of \hyphenchar	110		
2017-01-24 ltoutenc.dtx v2.0c	General: Added TU specific commands in ASCII range pr/4500	435	2017-03-27 ltdefns.dtx v1.5b	\@dischyp: Define \@dischyp after \-	110		
2017-01-24 ltoutenc.dtx v2.0d	General: Declare TU composites for i and j	435	2017-03-28 ltluatex.dtx v1.1e	General: glyph_stream_provider added	64		
2017-02-12 ltoutenc.dtx v2.0e	Make \textasteriskcentered U+2217 not U+204E	435	2017-03-29 ltboxes.dtx v1.3a	\arrayparboxrestore: Reset \lineskiplimit	735		
2017-02-18 ltluatex.dtx v1.1c	TeX ligature syntax for xetex and luatex reversed	435	2017-04-05 ltoutenc.dtx v2.0i	\DeclareTextCompositeCommand: Declare accent command if not already declared when declaring a composite.	410		
	\new_attribute: Parameterize count used in tracking	58	2017-04-10 lplain.dtx v2.3c	\newlanguage: Correction to code to skip write18 in luatex	18		
	\new_bytecode: Parameterize count used in tracking	59					
	\new_chunkname: Parameterize count used in tracking	59					
	\new_whatsit: Parameterize count used in tracking	59					

2017-04-11 ltoutput.dtx v2.4a		2018-05-08 ltclass.dtx v1.2i
\newpage: account for the depth of the last row of the page	998	\pkgcls@parse@date@arg: Make suspicious rollback a warning not error: github issue 43
2017-12-17 ltoutput.dtx v1.4b		919
\@addtonextcol: fix doc guards . . .	1031	2018-05-11 ltfinal.dtx v2.18
2018-01-06 ltdefns.dtx 1.5c		General: Make invalid UTF-8 also safe, for legacy filesystem encodings
\@ifundefined: Avoid defining undefined commands to \relax	106	1070
2018-02-18 ltclass.dtx v1.2d		2018-05-29 ltclass.dtx v1.2j
\@ifl@ter: Added 0 up front to make bad data come out as 0.	884	\endfilecontents: use \csname not \@undefined
General: Introduce rollback concept	915	911
2018-03-08 ltcnts.dtx v1.1k		2018-08-11 ltoutenc.dtx v2.0j
\@ifbothcounters: Interface added . . .	452	General: Provide \guillemetleft and \guillemetright
\@removefromreset: Interface added . . .	452	423, 429, 438
\counterwithin: Interface added	453	2018-08-18 ltluatex.dtx v1.1h
2018-03-24 ltclass.dtx v1.2e		General: append_to_vlist_filter is exclusive
\pkgcls@use@this@release: Use full file name for old release	921	64
2018-03-25 ltfinal.dtx v2.1a		2018-08-24 ltfinal.dtx v2.1f
General: default to UTF-8	1068	\document@default@language: Add to latexrelease (github/68)
\UseRawInputEncoding: Macro added	1069	1067
2018-03-27 ltclass.dtx v1.2f		2018-09-02 ltsect.dtx v1.1b
\endfilecontents: Use full file name for old release	910	\@dottedtocline: Prevent protrusion (https://tex.stackexchange.com/q/172785/10109)
2018-04-06 ltfinal.dtx v2.1b		814
\UseRawInputEncoding: Undo changes to \DeclareFontEncoding@ and definition of \DeclareUnicodeCharacter	1069	2018-09-24 fontdef.dtx v3.0b
2018-04-07 ltfinal.dtx v2.1c		General: Start LR-mode if necessary (git/49)
\UseRawInputEncoding: Undefine \inputencodingname	1069	616
2018-04-08 ltclass.dtx v1.2g		2018-09-24 ltmath.dtx v1.2b
\@ifl@ter: Strip leading spaces from dates.	884	\smash: Start LR-mode if necessary (git/49)
2018-04-08 ltclass.dtx v1.2h		697
\onefilewithoptions: Pass expanded date	917	\phantom: Start LR-mode if necessary (git/49)
2018-04-08 ltfinal.dtx v2.1d		696
General: Delay full UTF-8 handling to \everyjob	1070	2018-09-24 ltspace.dtx v1.3h
2018-04-11 ltcnts.dtx v1.1l		\enspace: Start LR-mode if necessary (git/49)
\counterwithin: Correct default (issue/38)	453	377
2018-05-02 ltluatex.dtx v1.1g		\leavevmode@ifvmode: Macro added (git/49)
General: find_sfd_file removed	63	377
finish_syntex_callback added	64	2018-09-26 ltdefns.dtx v1.5e
glyph_not_found added	64	\renew@command: Always explicitly generate a space after the csname and not rely on \noexpand to save tokens (git/41)
read_sfd_file removed	63	86
		2018-09-26 ltmiscen.dtx v1.1n
		\writefile: Sometimes mask the endline char when writing to files (github/73)
		677
		\add@percent@to@temptokena: Sometimes mask the endline char when writing to files (github/73)
		676
		\protected@file@percent: Sometimes mask the endline char when writing to files (github/73)
		676
		2018-09-26 ltsect.dtx v1.1c
		\addcontentsline: Sometimes mask the endline char when writing to

files (github/73)	812	2019-02-07 ltfssbas.dtx v3.2b
2018-10-10 ltspace.dtx v1.3i		\define@newfont: Changed wording of warning (github/107) 473
\@esphack: Don't introduce breakpoints if @nobreak is true and after sections	368	
2018-10-11 ltmissen.dtx v1.1o		2019-06-18 ltluatex.dtx v1.1j
\@0@svrb: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69)	690	General: finish_syncTeX_callback renamed finish_syncTeX 64
\@setupverbvisibleSpace: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69)	688	font_descriptor_objnum_provider added 64
\@verbvisibleSpacebox: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69)	689	make_extensible added 64
\@asciispace: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69)	688	new_graf added 64
\@verbatim*: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69)	689	page_objnum_provider added 64
\@verbvisibleSpace: Provide \verbvisibleSpace such that it is usable in normal text (github/70)	688	process_pdf_image_content added 64
Provide visible space in \verb* also for XeTeX and LuaTeX (github/69)	688	wrapup_run added 64
2018-10-21 ltluatex.dtx v1.1i		2019-07-01 ltclass.dtx v1.3a
new_luafunction: Function added	60	\endfilecontents: Support UTF8 and spaces in filecontents environment file name 907
2018-11-09 ltbibl.dtx LaTeX2e		2019-07-01 ltfssini.dtx v3.1c
\bibliography: Zap spaces in the argument as BibTeX doesn't support them (github/88)	844	General: Explicitly set some defaults 595
2018-11-18 ltoutenc.dtx v2.0k		2019-08-22 ltxref.dtx v1.11
General: Provide \Hwithstroke and \hwithstroke	440	\labelformat: Commanded moved from variorref.sty 669
2018-11-19 ltoutenc.dtx v2.0k		\Ref: Commanded moved from variorref.sty 669
General: Added \Hwithstroke and \hwithstroke	422	\refstepcounter: Allow \p@... to have an argument 669
2018-11-28 ltoutput.dtx v1.4d		2019-08-27 fontdef.dtx v3.0c
\@combinedblfloats: Unbox \@outputbox to preserve boxing level (github/94)	1017	General: Various commands made robust throughout the file 609
2018-12-30 lttab.dtx v1.1p		2019-08-27 ltboxes.dtx v1.3b
\@tabclassz: Add extra \hskip to guard against an \unskip at the start of a c-column cell (gh/102)	762	General: Various commands made robust 725
2019-02-07 ltfilehook.dtx v1.1o		2019-08-27 ltcass.dtx v1.3b
\unqu@tefilef@und: Expand \@filef@und before executing second argument (github/109)	943	\endfilecontents: Make various commands robust 907
2019-02-07 ltfssini.dtx v1.1o		2019-08-27 ltdefns.dtx v1.5f
\@swaptwoargs: Helper macro added	398	General: Make various commands robust 111
		\MakeRobust: Make the assignments global as we may need to apply them inside a group 92
		2019-08-27 ltfilehook.dtx v1.2b
		\unqu@tefilef@und: Make command robust 943
		2019-08-27 ltfssini.dtx v1.2b
		\IfFileExists: Make command robust 394

2019-08-27 ltfssbas.dtx v3.2d		2019-09-09 ltfssdcl.dtx v3.0s
General: Make various commands robust	459	\DeclareMathSymbol: Allow definition if the math symbol was a command already robust
2019-08-27 ltfsdcl.dtx v3.0s		560
\DeclareMathAccent: Make math accents robust	558	2019-09-11 ltclass.dtx v1.3c
\set@mathdelimter: Make math delimiters robust	564	\endfilecontents: Support optional argument for filecontents
2019-08-27 ltfsini.dtx v3.1d		907
General: Make various commands robust	570	2019-09-14 ltfinal.dtx v2.1h
2019-08-27 ltidxglo.dtx v1.1f		\@ucclist: Expand UTF8 chars when case changing (github/177) 1074
General: Make \index and \glossary robust	839	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	457	669
2019-08-27 ltlogos.dtx v1.1j		2019-09-21 fontdef.dtx v3.0d
\TeX: Make \TeX command robust ..	379	General: Distangle alias (gh/184) 611, 615
2019-08-27 ltmath.dtx v1.2c		2019-10-02 ltxpl.dtx v0.0
General: Make various commands robust	693	General: Initial version
2019-08-27 ltmiscen.dtx v1.1p		74
General: Make various commands robust	671	2019-10-02 ltfinal.dtx v2.2
\begin: Make command robust ..	679	General: Load ltxpl
\end: Make command robust	681	1077
2019-08-27 ltoutput.dtx v1.4e		2019-10-02 lthuataex.dtx v1.1k
\@begindvibox: Make \AtBeginDvi robust	996	General: linebreak_filter is exclusive 64
2019-08-27 ltpage.dtx v1.0l		mlist_to_hlist is exclusive
General: Make various commands robust	869	64
2019-08-27 ltpictur.dtx v1.1m		process_rule is exclusive
General: Make various commands robust	767	64
2019-08-27 ltsect.dtx v1.1d		2019-10-07 ltab.dtx v1.1q
General: Make various commands robust	803	\extracolsep: This needs to expand 754
2019-08-27 ltspace.dtx v1.3j		2019-10-11 ltfiles.dtx v1.2c
General: Make various commands robust	360	\set@curr@file: Remove one brace group
2019-08-27 ltab.dtx v1.1q		393
General: Make various commands robust	742	2019-10-11 lfsstrc.dtx v3.0l
Remove several unnecessary \gdef definitions	742	\@font@aliasinfo: Added 'alias' size function
2019-08-30 lterror.dtx v1.2q		531
\conditionally@traceoff: Macro added	338	2019-10-18 ltclass.dtx v1.3d
\conditionally@traceon: Macro added	338	\load@onefilewithoptions: Initialize \...-h@k only when loading the package or class (gh/198)
		899
		2019-10-22 lthuataex.dtx v1.1j
		General: page_objnum_provider and process_pdf_image_content classified data
		64
		2019-10-25 ltmiscen.dtx v1.1q
		\add@percent@to@temptokena: Allow unbalanced conditionals in #1 (gh/202)
		676
		2019-10-26 ltfiles.dtx v1.2d
		\@iffileonpath: quote on openin ... 397
		\IfFileExists: don't quote name .. 394
		\IfFileExists@: quote on openin ... 395
		\set@curr@file: remove quotes ... 393
		2019-11-01 ltdirchk.dtx v1.3a
		\filename@parse: take last . not first 12
		2019-11-02 ltmiscen.dtx v1.1s
		\@centercr: Make \@centercr robust (gh/203)
		683

2019-11-02 ltspace.dtx v1.3k		2020-01-20 ltoutenc.dtx v2.0n
\@normalcr: Make also \@normalcr robust 364		General: fix for gh/251 418
2019-11-09 ltfiles.dtx v1.2e		2020-01-22 lttextcomp.dtx v1.0b
\set@curr@file: expand and \string before removing quotes 393		\@tc@subst: The overall default is \textcompsubstdefault not \substddefault 634
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) 676		General: Load t1enc.def last (gh/255) 600
2019-11-11 ltfiles.dtx v1.2f		2020-01-25 ltoutenc.dtx v2.0m
\@iffileonpath: make \@filef@und match quoting used on \openin . 397		General: Load each encoding file only once (gh/255) 447
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) 448		\endfilecontents: Fix typo in error message 909
2019-11-29 ltclass.dtx v1.3e		2020-01-28 ltclass.dtx v1.3h
\@pr@videopackage: Protect package info text (gh/52) 887		\endfilecontents: Allow spaces in option string and display only unknown options not the whole option list (gh/256) 908
2019-12-17 fontdef.dtx v3.0e		2020-01-31 ltvers.dtx v1.1e
\mddefault: Set \bfdefault to "b" 603		General: Allow for upcoming format as pre-release 0 39
\shapedefault: Set \shapedefault explicitly to "n" 604		2020-02-02 ltluatex.dtx v1.1l
\updefault: Set \updefault to "up" 603		General: Add reverselist callback type 67 glyph_info added 64
2019-12-17 lftntcmd.dtx v3.4c		page_order_index added 64
\textssc: Macro added 625		post_linebreak_filter is reverselist 64
2019-12-17 lfssbas.dtx v3.2e		create_callback: Provide proper \fallbacks for user-defined callbacks without user-provided default handler 68
\usefont: Don't call \fontseries or \fontshape 468		2020-02-05 lfssini.dtx v3.1g
2019-12-17 lfssini.dtx v3.1e		\DeclareFontSeriesDefault: Clarified error text 572
General: Provide custom series settings a la mweights 571		Corrected misspelled csname (gh/264) 572
\DeclareEmphSequence: Provide \emph sequences 589		2020-02-05 lttextcomp.dtx v2.0n
2019-12-18 ltoutenc.dtx v2.0m		General: Changed the package default to info (gh/262) 651
General: Don't fake \textcompwordmark; take default from T1 instead 417		Ensure we are on a new format (gh/260) 651
\add@accent: Avoid code that breaks \accent 410		2020-02-07 lfssini.dtx v3.1h
2019-12-21 fontdef.dtx v3.0e		\symbol: XeTeX-specific version to avoid bug in maths mode. 592
General: Distangle alias (gh/184) 609–612		2020-02-10 lfssaxes.dtx v1.0c
2020-01-05 ltclass.dtx v1.3f		\fontseries: Switch \if@forced@series added 496
\endfilecontents: Support more write streams in LuaTeX gh/238 907		\fontseriesforce: Switch \if@forced@series added 496
2020-01-11 lfssini.dtx v3.1f		\if@forced@series: Switch \if@forced@series added 496
\rmfamily: Streamlined implementation with hook 584		
\ttfamily: Streamlined implementation with hook 584		
2020-01-20 lfssdcl.dtx v3.0t		
\set@mathdelimiter: fix for gh/251 564		

2020-02-10 ltfssini.dtx v3.1h	\@defaultfamilyhook: Add \@defaultfamilyhook to \normalfont (gh/269)	585	\prepare@family@series@update: Drop surplus “m” from \target@series@value (gh/291)	576
	\reset@font: Add \@defaultfamilyhook to \normalfont (gh/269)	593	\update@series@target@value: Drop surplus “m” from \reserved@d (gh/291)	576, 577
2020-02-10 lttextcomp.dtx v1.0c	General: Use \tabacckludge for tabbing where necessary (gh/271)	637	2020-02-27 ltdefns.dtx v1.5g	\gobblethree: Macro added
2020-02-11 fontdef.dtx v3.0g	General: Provide value for \@fontenc@load@list (gh/273)	600	2020-02-27 ltfssaxes.dtx v1.0d	\series@maybe@drop@one@m: Drop “m” in certain values from a fixed list (gh/293)
2020-02-11 ltfssini.dtx v3.1h	General: Provide default value for \@fontenc@load@list (gh/273)	595	\set@target@series: Drop “m” only in a specific set of values (gh/293)	499
2020-02-11 ltoutenc.dtx v2.0o	General: Update \@fontenc@load@list with option list (gh/273)	448	2020-02-27 ltfssbas.dtx v3.2g	\DeclareFontShape@: Only “m” if the series value is a member of a fixed list and issue warning if doing it (gh/293)
2020-02-14 ltpictur.dtx v1.1n	\linethickness: Suppress spaces following the declaration (gh/274)	772	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	\bfseries: Make the \ifx selection outside of \fontseries argument so that it is not done several times	579	2020-03-05 ltexpl.dtx v1.1	General: Load xparse.ltx if \NewDocumentCommand is not defined by expl3.ltx
	\mdseries: Make the \ifx selection outside of \fontseries argument so that it is not done several times	580	2020-03-06 ltboxes.dtx v1.3c	\clap: Macro \clap added
\prepare@family@series@update: No	series auto-update when forced (gh/277)	575	2020-03-07 lthuataex.dtx v1.1m	\remove_from_callback: Do not call callback.register for user-defined callbacks
Recognize current family if it is not a “meta” family and auto-update series using \bfdefault (gh/277)	575	2020-03-07 ltmath.dtx v1.2e	\negthickspace: Add amsmath math/text spacing commands to the kernel (gh/303)	
2020-02-18 ltmath.dtx v1.2d	\mathindent: Make \mathindent a skip register to match amsmath (gh/252)	706	2020-03-07 ltspace.dtx v1.3l	General: Moved \thinspace, \negthinspace and \, to ltmath.dtx (gh/303)
	\equation: Separate formula and eqn number by at least a space in fleqn option	707	2020-03-19 fontdef.dtx v3.0h	General: Support legacy use of \bfdefault and \mddefault (gh/306)
2020-02-20 ltclass.dtx v1.3j	\endfilecontents: Fix missing quotes around file name (gh/284)	909	2020-03-19 ltfssdcl.dtx v3.0u	\document@select@group: fix for (gnats/3357)
2020-02-24 ltfssbas.dtx v3.2f	\DeclareFontShape@: Drop surplus “m” in series when defining fontshape (gh/289)	460	2020-03-19 ltfssini.dtx v3.1k	\DeclareFontSeriesDefault: Support legacy use of \bfdefault and \mddefault (gh/306)
2020-02-25 ltfssini.dtx v3.1j	\bf@def@ult: Drop surplus “m” from \bfdef@ult and \mddef@ult (gh/291)	582	\maybe@update@bfseries@defaults: Support legacy use of \bfdefault and \mddefault (gh/306)	579

\mdseries: Support legacy use of \bfdefault and \mddefault (gh/306)	580	2020-04-21 ltab.dtx v1.1r \@itabcr: Support calc syntax (gh/152)	748
2020-04-06 ltfssini.dtx v3.1m \bf@def@ult: Hook added (gh/306)	582	\@yargarraycr: Support calc syntax (gh/152)	756
\maybe@update@bfseries@defaults: Hook added (gh/306)	579	2020-04-22 ltmiscen.dtx v1.1u \@csverb: Drop spaces before \verb delimiter (gh/327)	689
\maybe@update@mdseries@defaults: Hook added (gh/306)	580	2020-04-22 ltoutenc.dtx v2.0p General: y unicode value in tuenc.def	403
2020-04-07 ltclass.dtx v1.3k \IfFormatAtLeastTF: Macro added; also in rollback (gh/168)	883	2020-04-29 lttextcomp.dtx v1.0d General: Make all capital accents text commands for hyperref (gh/332)	637
\load@onefile@withoptions: Use different method to ignore unprocessed options (gh/22)	903	2020-05-02 ltfiles.dtx v1.2g \@include: Support spaces in filenames by enclosing the names of .aux-files in quotes (gh/217)	390
\ProcessOptions*: Use different method to ignore unprocessed options (gh/22)	892	\@includeonly: Get rid of leading and trailing spaces from the filename (gh/217)	388
\RequirePackageWithOptions: Use different method to ignore unprocessed options (gh/22)	895	Improved support for spaces in filenames (gh/217)	388
2020-04-09 ltfloat.dtx v1.2d \@textsubscript: Set non-zero baseline (gh/249)	835	Pass the filename to \@include by value instead of by reference (gh/217)	388
\textsubscript: Set non-zero baseline (gh/249)	834	2020-05-05 ltxref.dtx v1.1n \refstepcounter: record the counter name in \@currentcounter	668
2020-04-13 ltfssdcl.dtx v3.0v \process@table: Small update for speed.	549	2020-05-06 ltspace.dtx v1.3n General: Made softhyphen active in TU engines	378
2020-04-13 ltfssini.dtx v3.1n \init@series@setup: Handling \seriesdefault changes (gh/315)	578	2020-05-09 ltdefns.dtx v1.5j \@if@DeclareRobustCommand: Added \DeclareCommandCopy (gh/239)	101
\seriesdefault@kernel: Handling \seriesdefault changes (gh/315)	596	\@DeclareCommandCopy: Added \DeclareCommandCopy (gh/239)	98
2020-04-21 ltmath.dtx v1.2f \@yeqncr: Support calc syntax (gh/152)	704	2020-05-11 ltdefns.dtx v1.5j \@dischyp: Do not overwrite \- under LuaTeX	110
2020-04-21 ltmiscen.dtx v1.1t \@icentercr: Support calc syntax (gh/152)	684	2020-05-15 ltdefns.dtx v1.5g \typeout: Allow \par in the argument (gh/335)	80
2020-04-21 ltpictur.dtx v1.1o \@istackcr: Support calc syntax (gh/152)	772	2020-05-19 ltfssaxes.dtx v1.0e \series@maybe@drop@one@m: Need to use \edef (gh/336)	500
2020-04-21 ltspace.dtx v1.3m \@hspage: Support calc syntax (gh/152)	376	2020-05-19 ltfssini.dtx v3.2a \IfFontSeriesContextTF: Macros added (gh/335)	587
\@newline: Support calc syntax (gh/152)	365	2020-05-31 ltmiscen.dtx v1.1u \centering: Added \finalhyphendemerits setting (gh/247)	684
\@vspace@calcify: Support calc syntax (gh/152)	365		
\@vspacer: Support calc syntax (gh/152)	374		
\addvspace: Support calc syntax (gh/152)	371		

\raggedleft: Added \finalhyphendemerits setting (gh/247)	685	\date: Don't make the command \long (gh/354)	803
\raggedright: Added \finalhyphendemerits setting (gh/247)	685	2020-08-01 lltuatem.dtx v1.1p General: new_graf is exclusive	64
2020-06-04 ltexpl.dtx v1.2c General: Define a local version of some L ^A T _E X 2 _≤ basic macros to support package loading	74	2020-08-02 lltuatem.dtx v1.1q \newattribute: Move reset to 0 inside conditional	50
2020-06-04 ltfinal.dtx v2.2a General: Load ltexpl in ltdefns . . .	1077	\newluabytocode: Move reset to 0 inside conditional	53
2020-06-05 ltclass.dtx v1.3l \@currnamestack: Added \@expl@pop@filename@@	881	\newluachunkname: Move reset to 0 inside conditional	53
Added \@expl@push@filename@@ and \@expl@push@filename@aux@@	880	\newluafunction: Move reset to 0 inside conditional	52
2020-06-05 ltfiles.dtx v1.2h \document: Added hook to load l3backend code	382	\newwhatsit: Move reset to 0 inside conditional	52
2020-06-10 lltuatem.dtx v1.1n General: Define \@gobble/\@firstofone even for L ^A T _E X to allow early loading.	48	2020-08-08 ltclass.dtx v1.3m \endfilecontents: define \q@curr@file directly as the quotes have already been removed (gh/220)	909
2020-07-04 ltoutenc.dtx v2.0q General: Implement \remove@tlig in LuaT _E X without font reloading	435	2020-08-10 lltuatem.dtx v1.1r General: Load lltuatem Lua module during format building	53
2020-07-08 ltexpl.dtx v1.2d General: Add a last-minute hook for expl3	75	2020-08-15 ltpictur.dtx v1.2a \@defaultunitsset: Macro added . . .	769
2020-07-08 ltfinal.dtx v2.2b General: Add a last-minute hook for expl3	1066	2020-08-19 ltdefns.dtx v1.5k \@carcube: Made \long for \NewCommandCopy	83
2020-07-27 ltmath.dtx v1.2g \cases: Don't make the command \long (gh/354)	697	\robust@command@act: Made \robust@command@act (was \declare@command@copy) more generic	96
\matrix: Don't make the command \long (gh/354)	697	\ShowCommand: Added \ShowCommand (gh/373)	100
\pmatrix: Don't make the command \long (gh/354)	697	2020-08-19 ltexpl.dtx v1.2e General: Add	
2020-07-27 ltoutenc.dtx v2.0r \use@text@encoding: Don't make the command \long (gh/354)	412, 413	\@expl@cs@{thing}@spec@@N for \ShowCommand (gh/373)	78
2020-07-27 ltpage.dtx v1.0m \markright: Don't make the command \long (gh/354)	870, 871	Add \@expl@cs@to@str@@N and \@expl@str@if@eq@@nnTF for \NewCommandCopy (gh/239)	77
2020-07-27 ltpictur.dtx v1.1p \linethickness: Don't make the command \long (gh/354)	772	2020-08-20 lplain.dtx v2.3d \alloc@: Define \alloc@ in terms of \@alloc	22
2020-07-27 ltsect.dtx v1.1e \author: Don't make the command \long (gh/354)	803	2020-08-21 ltclass.dtx v1.3o General: Integration of new hook management interface	874
		2020-08-21 ltdefns.dtx v1.5l General: Integration of new hook management interface	80
		2020-08-21 ltdefns.dtx v1.5m \MakeRobust: Make \MakeRobust produce the same command	

structure as		2020-09-26 ltfinal.dtx v2.2j
\DeclareRobustCommand	92	General: Load first aid file if existing
2020-08-21 ltexpl.dtx v1.2d		1078
General: Dropped unused command	74	
2020-08-21 ltfiles.dtx v1.2i		
General: Integration of new hook		
management interface	380	\maybe@update@bfseries@defaults:
2020-08-21 ltfinal.dtx v2.2i		\bfdefault@previous not
General: Integration of new hook		\bfseries@previous (gh/395) ..
management interface	1060	579
2020-08-21 ltfsaxes.dtx v1.0g		\mdseries: \mddefault@previous not
General: Integration of new hook		\mdseries@previous (gh/395) ..
management interface	507	580
2020-08-21 ltfsini.dtx v3.2b		2020-10-01 ltclass.dtx v1.3r
\bf@def@ult: Integration of new hook		\opr@videopackage: Allow for package
management interface	582	substitution
\mdseries@defaults: Integration of		887
new hook management interface	585	2020-10-01 ltsect.dtx v1.1e
\maybe@update@bfseries@defaults:		\addcontentsline: add a fourth
Integration of new hook		argument for better hyperref
management interface	579	compatibility
\maybe@update@mdseries@defaults:		812
Integration of new hook		2020-10-04 ltfiles.dtx v1.2j
management interface	580	\include: Quotes around the aux file
\reset@font: Integration of new hook		name removed, they are not
management interface	593	needed and upset BibTeX
\rmfamily: Integration of new hook		(gh/400)
management interface	584	390
\ttfamily: Integration of new hook		2020-10-04 lthooks.dtx v1.0d
management interface	584	General: Definition \AddToHookNext
2020-08-21 ltmiscen.dtx v1.1v		was supposed to be for \AddToHook
General: Integration of new hook		vice versa (gh/401)
management interface	671	295
2020-08-21 ltoutput.dtx v1.4f		2020-10-08 ltclass.dtx v1.3s
\begin{dvbox}: Integration of new		\currnamestack: Added missing
hook management interface	996	2020/02/02 \IncludeInRelease ..
2020-08-23 ltxref.dtx v1.1o		880
\refstepcounter: add default		2020-10-11 ltclass.dtx v1.3t
definition of \currentcounter	668	\load@onefilewithoptions: Restore
2020-09-06 ltclass.dtx v1.3q		\currpkg@reqd after finished
\load@onefilewithoptions: Save		loading a package file (gh/408) ..
\currpkg@reqd so that we don't		901
lose track of package		2020-10-18 ltclass.dtx v1.3t
substitutions.	901	\PassOptionsToClass: Drop path
2020-09-06 ltdefns.dtx v1.5n		from \input@path (gh/414) ..
\char@if@alph: Macro added	109	888
2020-09-06 ltexpl.dtx v1.2f		2020-10-23 ltmiscen.dtx v1.1w
General: Add		\enddocument: Make
\expl@str@map@function@NN and		\enddocument/afteraux one-time
for \string@makeletter (gh/386)	78	673
2020-09-09 ltshipout.dtx v1.0b		\kernel@before@enddocument:
__shipout_picture_overlay:n		\enddocument should always start
Prevent overfull box warnings		out in vmode (gh/385)
(gh/387)	976	675
2020-11-09 ltmath.dtx v1.2h		2020-11-09 ltclass.dtx v1.3u
\char@grid@spaced@negmedspace and		\pkgcls@rollbackdate@error:
\negthickspace have been only in		Change help text because the
amsmath, so we need to undefine		package may have existed then —
for rollback (gh/423)		there is just no rollback data
		(gh/423)
2020-11-20 ltclass.dtx v1.3u		922
\currpath: Macro added		2020-11-09 ltmath.dtx v1.2h
		\char@grid@spaced@negmedspace and
		\negthickspace have been only in
		amsmath, so we need to undefine
		for rollback (gh/423)
		700
		2020-11-20 ltclass.dtx v1.3u
		\currpath: Macro added
		879

\@kernel@\currpathstack: Macro added	882	2020-12-04 ltfssaxes.dtx v1.0h General: Reorganized the rollback data	487
\load@onefile@withoptions: Copy option list to the requested package.	902	\fontseries: Distangle series and shape update (gh/444)	496
\PassOptionsToClass: Copy option list to the requested package. . .	889	\fontshape: Distangle series and shape update (gh/444)	504
\ProvidesPackage: Use string comparison instead of \ifx . . .	886	\fontshapeforce: Distangle series and shape update (gh/444)	504
2020-11-20 ltcmd.dtx v1.0a General: Initial version derived from xparse.dtx	113	2020-12-04 ltfssini.dtx v3.2f General: Adjust start values for series and shape (gh/444)	595
2020-11-20 ltfilehook.dtx v1.0d \unqu@tfilef@und: Move loading to \@input@file@exists@with@hooks and expand \@filef@und to avoid getting the wrong file name in the case of a substitution.	942	2020-12-10 ltbibl.dtx v1.1s \nocite: Delay any \nocite in the preamble instead of raising an error	845
2020-11-23 ltshipout.dtx v1.0d _shipout_execute_cont:: Check for both kernel and user hook (gh/431)	966	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)	468
_shipout_execute_main_cont:Nnnn: Check for both kernel and user hook (gh/431)	968	2020-12-14 ltclass.dtx v1.3v \@currnamestack: Removed \expl@@hook@curr@name@push@n .	880
2020-11-24 ltexpl.dtx v1.2g General: Support for roll forward (gh/434)	76–78	2020-12-18 ltexpl.dtx v1.2h \@kernel@after@enddocument@afterlastpage: Define kernel \enddocument hooks early	74
2020-11-24 ltfilehook.dtx v1.0d General: Support for roll forward (gh/434)	940	2020-12-22 ltfssaxes.dtx v1.0h \delayed@merge@font@series: Distangle series and shape update (gh/444)	498, 499
2020-11-24 lthooks.dtx v1.0f _hook_end_document_label_check:: Support for roll forward (gh/434)	233	\delayed@merge@font@shape: Distangle series and shape update (gh/444)	505
2020-11-24 ltshipout.dtx v1.0d General: Support for roll forward (gh/434)	980	2020-12-22 lfsstrc.dtx v3.0n \selectfont: Execute delayed series and shape updates (gh/444) . . .	512
\AtBeginDvi: Support for roll forward (gh/434)	979	2021-01-07 ltfilehook.dtx v1.0e General: Added rollback for this case to avoid spurious errors (part of gh/463)	954
2020-11-25 ltdefns.dtx v1.5o \carcube: Added missing latexrelease entry	83	\unqu@tfilef@und: Restore \CurrentFile(Path)(Used) after the input (gh/464)	943
2020-12-02 ltluatex.dtx v1.1s General: Fix return value of list callbacks	66	2021-01-07 lthooks.dtx v1.0h _hook_strip_double_slash:w: Assume hook name has at least three nonempty parts (gh/464) .	245
2020-12-03 lfsstrc.dtx v3.0m \selectfont: Install a hook in \selectfont (gh/444)	513	_hook_t1_set:cn: Manually define some l3tl commands to work around expl3 changes	220
2020-12-04 ltfilehook.dtx v1.0d \undeclare@file@substitution: Don't drop file substitution commands on rollback	945		

2021-01-08 ltshipout.dtx v1.0f __shipout_execute_cont:: Added another kernel hook for more flexibility (cf https://github.com/pgf-tikz/pgf/issu... 966)	2021-02-10 ltfloat.dtx v1.2e \@footnotetext: Explicitly run \par at the end of footnote text in preparation for paragraph hooks 835
2021-01-10 ltshipout.dtx v1.0g \@kernel@after@shipout@background: Internal hook \@kernel@after@shipout@background added 970	\document@select@group: fix for (gh/501) 544
\RawShipout: Macro added 969	2021-02-16 ltfloat.dtx v1.2f \footref: \footref added 838
2021-01-19 ltshipout.dtx v1.0h __shipout_run_firstpage_hook:: Handling of firstpage hook altered 970	2021-02-17 ltoutenc.dtx v2.0t General: Adjust values for \textasteriskcentered To match TS1 definition (gh/502) 440
2021-01-21 ltclass.dtx v1.3w \@kernel@currpathstack: Add empty entry for latexrelease 882	Special definition for \textasteriskcentered when missing in TS1 (gh/502) 431
2021-01-21 ltexpl.dtx v1.3a General: Move xparse rollback code to ltcmd.dtx 77	2021-02-18 ltclass.dtx v1.3x \@fileswithoptions: save raw class option list (gh/85) 897
2021-01-21 ltfinal.dtx v2.2l General: Load glyptounicode.tex for pdfTeX 1068	\@remove@eq@value: macro added (gh/85) 890
2021-01-22 ltshipout.dtx v1.0i __shipout_finalize_box:: Add pre_shipout_filter Lua callback 965	\@use@option: value from unused option list (gh/85) 893
2021-01-24 ltexpl.dtx v1.3a General: Define expl3 hooks conditionally 74	\OptionNotUsed: value from unused option list (gh/85) 890
2021-01-31 ltfilehook.dtx v1.0f \@curr@file@reqd: set \protect to \string gh/481 946	\PassOptionsToClass: save raw option lists (gh/85) 888
2021-02-03 ltfloat.dtx v1.2e \@savemarbox: Explicitly end with \par (gh/489) 829	2021-02-19 ltoutenc.dtx v2.0u General: Add \textnonbreakinghyphen, \textfiguredash and \texthorizontalbar (gh/404) 420, 424, 439
2021-02-04 ltboxes.dtx v1.4b \color@endbox: Always add the color groups (gh/488) 729	2021-02-25 ltfinal.dtx v2.2m General: Improve speed of ToUnicode everyjob loading code 1068
2021-02-08 ltfilehook.dtx v1.0g \unqu@tefilef@und: Undo the internal for robust \InputIfFileExists in rollback (gh/494) 944	2021-03-03 ltclass.dtx v1.3y \endfilecontents: Fix overwrite check for files with UTF-8 (gh/415) 909
2021-02-08 ltmiscen.dtx v1.1y \end: Undo the internals for robust \begin and \end in rollback (gh/494) 682	2021-03-05 ltclass.dtx v1.3z \ProcessOptions*: modify so braces to not give errors (gh/513) 891
2021-02-10 ltboxes.dtx v1.4b \@mpfootnotetext: Explicitly run \par in support for paragraph tagging 737	2021-03-12 ltfiles.dtx v1.2k \IfFileExists@: Allow unbalanced conditionals (gh/530) 395
	2021-03-18 ltcmd.dtx v1.0b General: Use \NewModuleRelease. . 113
	2021-03-18 ltfilehook.dtx v1.0h __filehook_file_pop_assign:nnnn: Define \g_@_input_file_seq to avoid losing data when rolling back. 940

2021-03-18 ltfssaxes.dtx v1.0i		2021-04-20 ltexpl.dtx v1.3c	
General: Fix rollforward definition.	497	\@kernel@after@enddocument@afterlastpage:	
2021-03-18 ltfssini.dtx v3.2g		Don't empty kernel hooks on	
General: Add legacy hook definitions		rollback	74
for rollback.	585	2021-04-20 ltfilehook.dtx v1.0i	
2021-03-18 lthooks.dtx v1.0i		\@curr@file@reqd: Make expand to	
__hook_end_document_label_check::		a string (tracks change in	
Only add top-level if not already		l3kernel)	946
there.	233	2021-04-26 ltfssbas.dtx v3.2i	
Remove the (empty) "top-level"		\usefont: Unconditionally switch to	
from \@currnamestack.	234	the requested font face (gh/444)	468
General: Use \NewModuleRelease.	218	2021-04-26 ltfssini.dtx v3.2h	
2021-03-18 ltvers.dtx v1.1f		\reset@font: Unconditionally switch	
\@check@IncludeInRelease: Add		to the requested font face	
support for usage in		(gh/444)	593
\NewModuleRelease	40	2021-04-26 ltfsstrc.dtx v3.0o	
\new@moduledate: Added		\selectfont: Unset the forced series	
\NewModuleRelease.	41	boolean when reaching	
2021-03-19 lttextcomp.dtx v1.0e		\selectfont (gh/444)	513
General: Use \NewModuleRelease . . .	631	2021-04-29 lthooks.dtx v1.0m	
2021-03-26 ltplain.dtx v2.3e		\ActivateGenericHook: Add	
\@unused: Allocate \@inputcheck and		\ProvideHook etc.	291
\@unused early so that they are		2021-04-29 ltoutenc.dtx v2.0v	
before expl3 allocates more		General: Add composites for	
streams (gh/538)	24	\ae/\AE/\æ/\鏗 (gh/552)	444
2021-03-27 ltclass.dtx v1.4a		2021-05-18 ltclass.dtx v1.4b	
\@currnamestack: Do not completely		\@raw@classoptionslist: Initialise to	
roll back if expl3 is loaded. . .	881	\relax to match	
2021-04-16 ltvers.dtx v1.1g		\@classoptionslist	879
\new@moduledate: \NewModuleRelease		2021-05-24 ltcmd.dtx v1.0e	
with the same arguments as		General: Use \msg_. . . instead of	
\IncludeInRelease.	41	__kernel_msg.	113
2021-04-17 ltfiles.dtx v1.2m		2021-05-24 ltcmdhooks.dtx v1.0b	
\@kernel@after@begindocument:		General: Use \msg_. . . instead of	
Move		__kernel_msg.	302
\@kernel@before@begindocument		2021-05-24 ltfilehook.dtx v1.0k	
and		General: Use \msg_. . . instead of	
\@kernel@after@begindocument		__kernel_msg.	938
init earlier so that other modules		2021-05-24 lthooks.dtx v1.0n	
can write to the hooks	385	General: Use \msg_. . . instead of	
2021-04-18 ltluatex.dtx v1.1t		__kernel_msg.	218
General: input_level_string added . .	64	2021-05-24 ltpara.dtx v1.0g	
2021-04-18 ltplain.dtx v2.3f		General: Use \msg_. . . instead of	
\loggingall: 3	34	__kernel_msg.	349
Drop pre- ϵ -TeX support	34	2021-05-26 ltdefns.dtx v1.5p	
\tracingnone: 3	35	\MakeRobust: Normalize error message	
Drop pre- ϵ -TeX support	35	in \MakeRobust	92
2021-04-19 ltcmd.dtx v1.0d		2021-06-03 ltclass.dtx v1.4c	
__cmd_cmd_type_cases:Nnnnn:		\@kernel@currpathstack: Take care	
Renamed		of \@kernel@currpathstack when	
__cmd_cmd_if_xparse:NTF to		rolling back/forward.	882
__kernel_cmd_if_xparse:NTF for		2021-06-04 ltcmd.dtx v1.0f	
cross-module usage	180	General: Normalize various error	
		messages	182

2021-06-05 ltmiscen.dtx v1.1z		2021-07-22 lthooks.dtx v1.0o	
\@sverb: Normalize error message .	689	__hook_gremove_code:n: Do not	
2021-06-06 ltclass.dtx v1.4c		queue removals (gh/625)	246
\@loadwithoptions: handle raw		__hook_prop_gput_labeled_do:Nnnn:	
options for gh/580	895	Do not queue removals (gh/625)	235
\load@onefile@withoptions: Copy		2021-07-23 ltclass.dtx v1.4e	
raw options for gh/580	902	\load@onefile@withoptions: Make	
\PassOptionsToClass: apply		class/name/after a one-time hook	903
\expandafter to raw options for		Make class/name/before a one-time	
gh/580	889	hook	902
2021-06-09 ltclass.dtx v1.4b		Make package/name/after a	
\endfilecontents: Use		one-time hook	903
\@latex@note@no@line to display		Make package/name/before a	
the information	909	one-time hook	902
2021-06-09 lterror.dtx v1.2r		2021-07-23 ltfiles.dtx v1.2n	
\@latex@note@no@line: Macros		\@include: Make include/name/after	
added	333	a one-time hook	390
2021-06-09 ltssbas.dtx v3.2j		Make include/name/before a	
\DeclareFontShape@: Improve		one-time hook	390
information message	460	Make include/name/end a one-time	
2021-07-08 ltcnts.dtx v1.1m		hook	390
\counterwithin: New implementation		2021-07-27 lthooks.dtx v1.0o	
for \counterwithout and		\ClearHookNext: Macro added	292
\counterwithin	452	\hook_gclear_next_code:n: Macro	
2021-07-11 lterror.dtx v1.2s		made public	277
\PackageNoteNoLine: Provide		2021-07-28 ltsect.dtx v1.1f	
\ClassNote and \PackageNote .	332	\contentsline: Pick up four	
2021-07-12 ltclass.dtx v1.4d		arguments (gh/633)	813
\@fileswithoptions: add		2021-07-30 lcmd.dtx v1.0d	
\unexpanded	898	__cmd_cmd_type_cases:Nnnnn:	
2021-07-16 lplain.dtx v2.3g		Added	
General: Use 2 as default value for		\@C_cmd_type_cases:NnnnnF for	
\tracinglostchars	25	\NewCommandCopy and	
2021-07-19 ltclass.dtx v1.4e		\ShowCommand support	180
\@classoptionslist: Drop		2021-07-31 ltoutput.dtx v1.4e	
\@onlypreamble	879	\f1@tracemessage: Enable display	
\@ifclasslater: Drop		when doing \tracefloatvals	1043
\@onlypreamble	883	2021-07-31 ltoutput.dtx v1.4g	
\@ifclassloaded: Drop		\ShowFloat: Macro added	1041
\@onlypreamble	883	2021-08-02 lthooks.dtx v1.0o	
\@ifclasswith: Drop		\ActivateGenericHook: Change	
\@onlypreamble	885	name	291
\@ifl@ter: Drop \@onlypreamble .	884	\DisableGenericHook: Change name	291
\@pkextension: Drop		2021-08-07 lcmd.dtx v1.0g	
\@onlypreamble	879	__cmd_add_grabber:N: Replicate	
\@optionlist: Drop \@onlypreamble .	883	argument processors for all	
\@unusedoptionlist: Drop		embellishments (gh/639)	138
\@onlypreamble	879	__cmd_add_type_E:w: Replicate	
\IfFormatAtLeastTF: Drop		argument processors for all	
\@onlypreamble	883	embellishments (gh/639)	136
2021-07-20 lcmdhooks.dtx v1.0c		2021-08-08 ltfinal.dtx v2.2p	
__hook_patch_DeclareRobustCommand:Nnn:		\IfPDFManagementActiveTF: Default	
Use \robust@command@chk@safe		definition added (gh/640)	1078
before \@if@newcommand.	307		

2021-08-10 ltvers.dtx v1.1h \@check@IncludeInRelease: Add error to aid debugging	40	2021-08-30 ltcmd.dtx v1.0h General: Added support for \NewCommandCopy	143
2021-08-11 ltluatex.dtx v1.1u General: Define missing local function	55	Added support for \ShowCommand	148
2021-08-20 lterror.dtx v1.2t \@badend: Improve \@badend error message (gh/587)	336	2021-09-03 ltoutput.dtx v1.4h \ShowFloat: Renamed, original name never distributed	1041
2021-08-20 lthooks.dtx v1.0p General: Added deprecation warnings for old generic hook commands (gh/638)	294	2021-09-06 ltfinal.dtx v2.2q \@uc1clist: Correctly upper and lowercase \ij and \IJ (gh/658)	1076
Documentation updates for generic hook commands (gh/638)	193	2021-09-06 lthooks.dtx v1.0r __hook_if_execute_immediately:n: Macro added (gh/606)	283
Renames of generic hook commands (gh/638)	229	__hook_prop_gput_labeled_do:Nnnn: Use dedicated conditional (gh/606)	235
Section on generic hooks added (gh/638)	209	__hook_use_once_clear:n: Clean up after \UseOneTimeHook (gh/606)	283
2021-08-25 ltclass.dtx v1.4f General: Standardise generic hook names (gh/648)	874	\hook_use_once:nnw: Clean up after \UseOneTimeHook (gh/606)	281
\load@onefile@withoptions: Declare non-generic package and class hooks	904	2021-09-10 ltfssini.dtx v3.2i \bfseries: Do delayed changes to \bfdefault in a separate macro for better reuse (gh/663)	579
2021-08-25 ltcmdhooks.dtx v1.0d __hook_try_put_cmd_hook:w: Simplify generic hook detection .	304	\DeclareFontSeriesDefault: Do delayed changes to \bfdefault or \mddefault first (gh/663)	572
2021-08-25 ltfilehook.dtx v1.0l \unqu@tefilef@nd: Declare non-generic file hooks	943	\maybe@update@bfseries@defaults: Do delayed changes to \bfdefault in a separate macro for better reuse (gh/663)	579
2021-08-25 ltfiles.dtx v1.2o \@include: Declare non-generic include hooks	391	\maybe@update@mdseries@defaults: Do delayed changes to \mddefault in a separate macro for better reuse (gh/663)	580
Standardise generic hook names (gh/648)	390	\mdseries: Do delayed changes to \mddefault in a separate macro for better reuse (gh/663)	580
2021-08-25 lthooks.dtx v1.0p __hook_try_declaring_generic_next_hook:nn: Standardise generic hook names (gh/648)	239	2021-09-12 lfntcmd.dtx v3.5a \check@nocorr@: use \unexpanded to make # safe	626
2021-08-27 ltcmd.dtx v1.0h \NewDocumentEnvironment: Check for end-of-environment command .	188	2021-09-12 ltoutenc.dtx v2.0w General: Move zero skip between i and j for hyphenation (gh/658)	421
2021-08-27 ltfilehook.dtx v1.0l __filehook_file_pop_assign:nnnn: Internal message name changes .	940	2021-09-18 ltpara.dtx v1.0i \para_end@: Use skip rather than kern as guard.	354
__filehook_file_subst_cycle_error:cN: Use \msg_... not __kernel_msg_...	951	2021-09-26 ltfssdcl.dtx v3.0x \c@localmathalphabets: Counter added for (gh/676)	543
2021-08-27 lthooks.dtx v1.0q General: Internal message name changes	287	\document@select@group: Test if we should freeze the version (gh/676)	543
2021-08-27 ltpara.dtx v1.0i General: Internal message name changes	356	\f@freeze@math@version: Macro added for (gh/676)	546

2021-09-28 ltcmdhooks.dtx v1.0e __hook_make_prefixes:w: Make patching of commands a global operation (gh/674) 312	2021-11-30 ltclass.dtx v1.5a \@onefilewithoptions@clashchk: Separated out from \@onefilewithoptions 901
__hook_patch_retokenize:Nnnn: Make patching of commands a global operation (gh/674) 320	2021-11-30 ltexpl.dtx v1.3d \skipeval: Moved over from xfp package (gh/711) 78
2021-09-28 lthooks.dtx v1.0s __hook_if_usable_use:n: Correct usage of older \@@_if_file_hook:wTF (gh/675) 281	2021-12-02 ltcmd.dtx v1.0i __cmd_run_code:: Correct defaults for optional arguments in end-of-environment code (gh/712) 122
__hook_try_declaring_generic_hook_split:nNnnn: Correct usage of older \@@_if_file_hook:wTF (gh/675) 240	__cmd_start_aux:ccnnnn: Correct defaults for optional arguments in end-of-environment code (gh/712) 121
2021-10-14 ltboxes.dtx v1.4c \@mpfootnotetext: Explicitly set \@currentcounter (gh/687) 737	2021-12-07 ltexpl.dtx v1.3d \skipeval: Added \dimeval and \skipeval (gh/711) 79
2021-10-14 ltfiles.dtx v1.2p \@include: Warn about use in preamble 390	2021-12-11 ltdirchk.dtx v1.3a General: Add comment lines into latex.ltx to indicate temp definitions that are later overwritten (gh/725) 2
2021-10-14 ltfloat.dtx v1.2g \@footnotetext: Explicitly set \@currentcounter (gh/687) 835	2021-12-12 ltoutenc.dtx v2.0y General: \DeclareUnicodeAccent now makes the encoding argument implicit (gh/253) 437
2021-10-14 ltmath.dtx v1.2j \@eqnse: Explicitly set \@currentcounter (gh/687) 703	Added \DeclareUnicodeCommand and \DeclareUnicodeSymbol (gh/253) 437
eqnarray: Explicitly set \@currentcounter (gh/687) 707	2021-12-27 lltuatem.dtx v1.1x \newprotectedluacmd: Macros added 52
2021-10-15 ltfssdcl.dtx v3.0y \DeclareMathVersion: Initialize variable for freezing math version (gh/676) 549	2021-12-28 ltexpl.dtx v1.3d \ExpandArgs: Added document level names for \exp_args:Nc and the like (gh/735) 79
2021-10-15 lltuatem.dtx v1.1v General: provide_charproc_data added 64	2021-13-31 ltcmd.dtx v1.0j __cmd_show:x: Make \ShowCommand stop for interaction 150
2021-10-16 ltoutenc.dtx v2.0x \add@accent: Dont set \spacefactor in math mode gh/643 410	2022-01-06 ltexpl.dtx v1.3e \ExpandArgs: Adjust document level names for \exp_args:Nc and the like (gh/735) 79
2021-10-19 ltpara.dtx v1.0k General: Remove content from \tex_everypar:D on rollback 357	2022-01-06 ltshipout.dtx v1.0l \AtBeginShipoutNext: Correctly simulate \AtBeginShipout and \AtBeginShipoutNext without extending the syntax 982
2021-10-20 ltcmdhooks.dtx v1.0f __hook_make_prefixes:w: Correct patching by expansion+redefinition when the macro contains a parameter token (gh/697). 308	\AtEndDvi: Correctly simulate \AtEndDvi without extending the syntax 981
2021-11-17 lltuatem.dtx v1.1w General: hpack_quality is exclusive 64 Never pass on true return values for list callbacks 66 vpack_quality is exclusive 64	2022-01-15 ltkeys.dtx v1.0b __keys_options_aux:n: Clear option list in end-of-package hook 927

2022-01-25 ltplain.dtx v2.3h	\obeyspaces: Provide redirection to support special use cases (gh/367)	29	2022-03-10 ltfilehook.dtx v1.0m	\@curr@file@reqd: Add \set@curr@file@nosearch for graphicx	946
2022-02-05 ltkeys.dtx v1.0b	.usage: Create properties in ltkeys	926	2022-03-18 ltclass.dtx v1.5b	\load@onefilewithoptions: Switch to \ProcessKeyOptions	900
2022-02-07 ltkeys.dtx v1.0c	.usage: Correct ..code property	926	2022-03-18 ltcmd.dtx v1.0l	__cmd_cmd_type_cases:Nnnnn: Fix \@@_cmd_type_cases:NnnnnF " prematurely expanding macros (gh/795)	180
2022-02-15 ltkeys.dtx v1.0c	\DeclareKeys: Expand module argument	930	2022-03-18 ltkeys.dtx v1.0f	__keys_options:n: Simplify to always cover global options	927
	\DeclareUnknownKeyHandler: Added \DeclareUnknownKeysHandler	930		__keys_options_global:n: Simplify to always cover global options	928
	\ProcessKeyOptions: Expand module argument	930		\ProcessKeyOptions: Remove \ProcessKeyPackageOptions	930
	\SetKeys: Expand module argument	931	2022-04-01 ltfiles.dtx v1.2q	\@include: Process some hooks is an include file is bypassed	391
2022-02-16 ltkeys.dtx v1.0d	\DeclareKeys: Allow for active characters in module argument	930	2022-04-01 lthooks.dtx v1.0t	\c_hook_generic_include//./end_tl: Support generic include//.../excluded hooks	245
	\DeclareUnknownKeyHandler: Allow for active characters in module argument	930	2022-04-03 ltfinal.dtx v2.2s	General: Integration of new mark management interface	1060
	Rename \DeclareUnknownKeysHandler to \DeclareUnknownKeyHandler	930		\opcol: Interface with new mark mechanism	1007
	\ProcessKeyOptions: Allow for active characters in module argument	930	2022-04-04 ltpage.dtx v1.0n	\markright: Interface with new mark mechanism	870, 871
	\SetKeys: Allow for active characters in module argument	931	2022-04-08 ltmath.dtx v1.2k	\openup: Make \protected (gh/123)	698
2022-02-19 ltcmd.dtx v1.0k	\IfBlankTF: Added \IfBlankTF and friends	190	2022-04-12 ltxref.dtx LaTeXe	\pageref: Macro reimplemented with a starred version	666
2022-02-20 ltfinal.dtx v2.2r	\@uclist: Use \expl@text@uppercase@n, removing local redefinition of \UTF@two@octets@noexpand	1074		\ref: Macro reimplemented with a starred version	666
	use \text_lowercase:n	1074	2022-04-12 ltxref.dtx v1.1p	General: Add starred variants for the ref commands	664
2022-02-21 ltkeys.dtx v1.0e	__keys_options_expand_module:Nn: Faster approach to module expansion	929		\Ref: Macro reimplemented with a starred version	669
2022-02-28 ltcmd.dtx v1.0k	General: Move latexrelease redefinitions from ltcmd.dtx	191	2022-04-21 ltfinal.dtx v2.2t	\@uclist: Support \noexpand in argument of \expl@text@uppercase@n	1074
2022-02-28 ltexpl.dtx v1.3f	General: Move latexrelease redefinitions from ltcmd.dtx	77	2022-05-06 ltmarks.dtx v1.0c	__mark_new_class:nn: Wrong command made \onlypreamble	855
2022-02-28 ltvers.dtx v1.1i	\new@moduledate: Detect a missing \IncludeInRelease{0000/00/00}.	41			
2022-03-10 ltbibl.dtx v1.1t	\cite: Ensure that an empty argument generates a warning (gh/790)	843			

2022-05-08 ltmath.dtx v1.2l		2022-06-20 ltclass.dtx v1.5c
General: Use consistent math styles under LuaTeX	705	\load@onefilewithoptions: Pass raw options to \ProcessKeyOptions
2022-05-08 ltshipout.dtx v1.0m		2022-06-20 ltkeys.dtx v1.0h
\@kernel@after@enddocument@afterlastpage: __keys_options_class:n: Use raw options data	928	
Handle case where shipout/lastpage is run too early (gh/813)	978	__keys_options_class:nnn: New function
__shipout_execute_main_cont:Nnnn: Handle case where shipout/lastpage is run too early (gh/813)	968	__keys_options_global:n: Use raw options data
2022-05-13 lthooks.dtx v1.0u		__keys_options_local::: Use raw options data
__hook_use_once_clear:n: Check if prop exists to avoid l3debug error	283	__keys_options_package:n: Use raw options data
2022-05-17 lthooks.dtx v1.0u		2022-06-22 ltkeys.dtx v1.0i .usage: Add ..notif property
__hook_initialize_hook_code:n: Refuse sorting one-time hooks (gh/818)	260	2022-06-30 ltfinal.dtx v2.2u \@cuclist: Add \AddToNoCaseChangeList
2022-05-17 ltmeta.dtx v1.0b		2022-06-30 ltfinal.dtx v2.2v \@cuclist: Just use \text_lowercase:n without \protectd@edf gh/881x
General: Default definition for targets added	359	2022-07-04 ltfinal.dtx v2.2w \@cuclist: Introduced \CaseSwitch, \DeclareCaseChangeEquivalent and \MakeTitlecase to support hooking into case changing gh/889
2022-05-27 ltfiles.dtx v1.2r		2022-07-04 ltfsbas.dtx v3.2k \frozen@everydisplay: Ignore spaces if necessary (gh/886)
\listfiles: Try saved version string, if ver@.. is \relax (gh/825)	401	\frozen@everymath: Ignore spaces if necessary (gh/886)
2022-05-27 ltoutenc.dtx v2.0z		2022-07-04 ltfsdcl.dtx v3.0z \freeze@math@version: Ignore spaces if necessary (gh/886)
General: Save the version string (gh/825)	448	2022-07-05 ltkeys.dtx v1.0i __keys_options_aux:n: Support \CurrentOption
2022-06-01 ltmarks.dtx v1.0d		__keys_options_class:nnn: Correct naming of raw class options storage
__mark_update_structure:nn: Extend the logic for detecting the marks in the box (gh/836)	857	2022-07-23 ltkeys.dtx v1.0j __keys_options_end:: Output ‘public’ package name in messages
General: Marks are kernel errors	862	__keys_options_loaded:nn: Output ‘public’ package name in messages
2022-06-02 ltfinal.dtx v2.2u		2022-08-07 lttextcomp.dtx v1.0g \DeclareEncodingSubset: Make global declaration (gh/905)
\@cuclist: Add \NoCaseChange	1075	2022-08-10 ltcmd.dtx v1.1a __cmd_arg_to_keyvalue:nn: New internal arg-to-keyval processor
2022-06-15 lthooks.dtx v1.0v		
__hook_activate_generic:n: Ensure that a newly activated generic hook gets its execution code set .	230	
2022-06-16 ltkeys.dtx v1.0g		
__keys_options_class:n: Better handling of option removal	928	
__keys_options_package:n: Better handling of option removal	929	
2022-06-19 ltkeys.dtx v1.0h		
__keys_options_class:n: Further work on handling of option removal	928	
__keys_options_package:n: Further work on handling of option removal	929	
__keys_options_package:nnn: New function	929	

__cmd_grab_D_long_obey_spaces_no_strip:w:declare_callback_rule: Add		
Add support for skipping brace	function	70
stripping		
.....	154
__cmd_grab_R_aux>NNnN: Track	remove_from_callback: Add rules for	
changes in D-type implementation	callback ordering	71
__cmd_grab_b_end:Nw: Track changes	2022-10-10 ltclass.dtx v1.5d	
in D-type implementation	\@unprocessedoptions: Use	
.....	\protected@edef.	907
General: New switch	\load@onefilewithoptions: Use	
.....	\protected@edef.	900
2022-08-13 lltuatax.dtx v1.1y	\PassOptionsToClass: Use	
General: shared_callbacks added	\protected@xdef.	889
.....	\ProcessOptions*: Use	
add_to_callback: Adapted code for	\protected@edef.	891
shared_callbacks	2022-10-20 ltclass.dtx v1.5e	
.....	\load@onefilewithoptions: Define	
remove_from_callback: Adapted code	key option handler in ltkeys	900
for shared_callbacks	
.....	\keys_options_aux:n: Define key	
mlist_to_hlist: Use shared_callback	option handler in ltkeys	927
system for	\keys_options_loaded:nn: Correct	
pre/post/mlist_to_hlist	an argument for a message	930
.....	2022-10-22 ltclass.dtx v1.5e	
2022-08-18 ltfilehook.dtx v1.0n	\fileswithoptions: Use	
\@disable@package@load@do: Inhibit	\protected@xdef.	897
checking the loaded version when	\use@option: Use \detokenize	893
package is load-disabled, and write	\ProcessOptions*: Use	
to the .log (gh/888)	\detokenize	891, 892
.....	2022-10-22 ltdefns.dtx v1.5r	
2022-08-20 ltoutput.dtx v1.4j	\expandtwoargs: Use	
\stockheight: Register added	\protected@edef.	89
.....	2022-10-22 ltkeys.dtx v1.0l	
\stockwidth: Register added	\keys_options_class:nnn: Correct	
.....	handling of unused option list	928
2022-08-21 ltkeys.dtx v1.0k	2022-10-26 ltfinal.dtx v2.2x	
\keys_options_loaded:nn: Correct	\@uclclist: Auto-detect babel	
error message	locale	1074
.....	Introduce optional argument for	
2022-09-03 ltmath.dtx v1.2m	case-changing commands	1074
\smash: Guard against reboxing in	Make case changing commands	
fractions (gh/517)	language-aware	1074
.....	2022-11-06 ltmiscen.dtx v1.2a	
2022-09-07 ltboxes.dtx v1.4d	\enddocument: Repeat release info at	
\@iiiminipage: Check for nested	the end (gh/944)	673
minipages and warn (gh/168)	2022-11-06 ltvers.dtx v1.1j	
.....	General: Repeat release info at the	
\if@in@minipage@env: Check for	end (gh/944)	39
nested minipages and warn	2022-11-08 ltshipout.dtx v1.0n	
(gh/168)	_shipout_execute_main_cont:Nnnn:	
.....	Add shipout hook (gh/920)	968
2022-09-17 ltfsdcl.dtx v3.1a	\shipout/lastpage: Add shipout hook	
\DeclareMathVersion: New logic for	(gh/920)	970
freezing math versions (gh/921)	2022-11-16 ltclass.dtx v1.5f	
.....	\endfilecontents: Do not show	
\freeze@math@version: New logic for	"current dir" in message (gh/917)	909
freezing math versions (gh/921)	
.....		
2022-09-20 ltfsdcl.dtx v3.1b		
\DeclareSymbolFont: Drop any		
surplus 'm' in series argument		
(gh/918)		
.....		
\SetSymbolFont: Drop any surplus 'm'		
in series argument (gh/918)		
.....		
2022-10-03 lltuatax.dtx v1.2a		
General: Add rules for callback		
ordering		
.....		
add_to_callback: Add rules for		
callback ordering		
.....		

	Introduce key 'nowarn' on filecontents (gh/958)	908, 909	2023-04-02 ltfilehook.dtx v1.0o \@set@curr@file@aux: Make \@set@curr@file@aux \long gh/942	948
2022-11-24	ltdefns.dtx v1.5s \DeclareEnvironmentCopy: Add \NewEnvironmentCopy, \RenewEnvironmentCopy, and \DeclareEnvironmentCopy (gh/963)	99	2023-04-06 ltcmdhooks.dtx v1.0g __hook_double_hashes_space:w: Add case for \c_@_hashes_t1 (hook-args).	313
	\ShowEnvironment: Added \ShowEnvironment	104	__hook_make_prefixes:w: Rename to \c_@_hashes_t1 (hook-args).	309
2022-11-28	ltspace.dtx v1.3o \@hspac: Support calc syntax correctly (gh/967)	376	\c_@_hook_hashes_t1: Rename to \c_@_hashes_t1 and add \c_@_hash_t1 (hook-args).	303
	\@vspace@\calcify: Support calc syntax without a group (gh/967)	365	2023-04-06 lthooks.dtx v1.1a __hook_activate_generic:n: Changes to add hook arguments (hook-args).	230
2022-11-29	ltcmd.dtx v1.1b __cmd_show:x: Add \@showenvironmentlisthook . . .	151	__hook_braced_real_loop:w: Macro added (hook-args).	254
2022-11-30	ltcmd.dtx v1.1b __cmd_show:x: Don't stop for the \begin part of an environment .	150	__hook_chk_args_allowed:nn: Macro added (hook-args).	238
2022-11-30	ltfinal.dtx v2.2y \@uclclist: Set \oe/\OE equal to act as a marker for babel	1075	__hook_clear_next:n: Changes to add hook arguments (hook-args).	278
2023-01-05	ltfiles.dtx v1.2s \document: \do now with default definition in the kernel (gh/975)	384	__hook_code_gset_aux:nnn: Macro added (hook-args).	249
2023-01-19	ltluatex.dtx v1.2b General: Remove unused local variable tex_setattribute	55	__hook_code_gset_auxi:een: Macro added (hook-args).	248
2023-01-30	ltpara.dtx v1.0l \g_para_standard_everypar_t1: Backout \parskip at top of minipage (gh/989)	350	__hook_cs_end:w: Macro added (hook-args).	253
2023-03-12	ltcmd.dtx v1.1c __cmd_copy_expandable:NnNNNNnnn: Distinguish (non-expandable) document commands starting with \@\@_start_expandable:nNNNNn .	145	__hook_cs_if_empty:c: Macro added (hook-args).	253
	__cmd_show:x: Distinguish (non-expandable) document commands starting with \@\@_start_expandable:nNNNNn .	149	__hook_gput_next_do:nn: Changes to add hook arguments (hook-args).	277
2023-03-22	ltspace.dtx v1.3p \samepage: Add \predisplaypenalty setting (gh/1022)	363	__hook_gremove_code:nn: Changes to add hook arguments (hook-args).	246
2023-03-28	ltclass.dtx v1.5g \IfFormatAtLeastTF: Added \IfFileAtLeastTF (gh/1015) . . .	883	__hook_if_execute_immediately:n: Changes to add hook arguments (hook-args).	283
2023-03-28	ltfinal.dtx v2.2z \@uclclist: Use groups for gh/1021	1075	__hook_init_structure:n: Changes to add hook arguments (hook-args).	226
2023-04-01	ltfssbas.dtx v3.2l \math@version: Reset frozen mathversion gh/1028	470	__hook_initialize_all:: Changes to add hook arguments (hook-args).	259
			__hook_initialize_hook_code:n: Changes to add hook arguments (hook-args).	260
			__hook_initialize_single:ccn: Changes to add hook arguments (hook-args).	263
			__hook_log:nN: Changes to add hook arguments (hook-args).	270

__hook_log_next_code:n: Changes to add hook arguments (hook-args).	274	(hook-args).	291
__hook_make_usable:nn: Changes to add hook arguments (hook-args).	224	\c_hook_??_parameter_tl: Token list added (hook-args).	255
__hook_new_reversed:nn: Add \hook_new_reversed_with_args:nn (hook-args).	227	\c_hook_nine_parameters_tl: Add auxiliary token lists (hook-args).	220
__hook_normalise_cs_args:nn: Macro added (hook-args).	250	\g_hook_replacing_stack_seq: Macro added (hook-args).	287
__hook_parameter:n: Macro added (hook-args).	255	\hook_gput_next_code:nn: Add \hook_gput_next_code_with_args:nn (hook-args).	276
__hook_post_initialization_defs:: Macro added (hook-args).	280	\hook_if_empty:n: Changes to add hook arguments (hook-args).	284
__hook_prop_gput_labeled_do:Nnnn: Add \hook_gput_code_with_args:nnn (hook-args).	235	\hook_new_pair_with_args:nnn: Add \hook_new_pair_with_args:nnn (hook-args).	227
Changes to add hook arguments (hook-args).	235, 236	\hook_use_once:nw: Add \hook_use_once:nw (hook-args).	281
Macro added (hook-args).	237	\NewMirroredHookPairWithArguments: Add \NewHookWithArguments (hook-args).	290
__hook_set_normalise_fn:nn: Macro added (hook-args).	251	\UseOneTimeHookWithArguments: Add \UseHookWithArguments (hook-args).	292
__hook_tl_set:cn: Clean-up unused variants (hook-args).	220	2023-04-11 ltfinal.dtx v2.3a \@uclclist: Use new generic mechanism to detect locale	1074
__hook_try_declaring_generic_hook:wn: Changes to add hook arguments (hook-args).	241	2023-04-13 ltcmd.dtx v1.1d __cmd_grab_v_group_end:: Set \tex_endlinechar:D earlier (gh/876).	161
__hook_use_i_delimit_by_s_mark:nw: Use standard naming scheme (hook-args).	220	2023-04-14 ltclass.dtx v1.5h \load@onefilewithoptions: Define \load@onefilewithoptions when in latexrelease (gh/992)	899
__hook_use_initialized:nnw: Add \hook_use:nnw (hook-args).	280	2023-04-15 ltplain.dtx v2.3i \extrafloats: Protect box 255 in lualatex gh/1041	22
__hook_use_once:nn: Changes to add hook arguments (hook-args).	282	unwind numexpr and ifnum nesting	22
__hook_use_once_clear:n: Changes to add hook arguments (hook-args).	283	2023-04-16 ltfinal.dtx v2.3a \BCPdata: Command added	1073
__hook_new:nn: Add \hook_new_with_args:nn (hook-args).	224	2023-04-16 lthooks.dtx v1.1b __hook_include_legacy_code_chunk:n: \@@_replacing_args_false: in \@@_include_legacy_code_chunk:n.	228
Update hook code after declaring.	224	2023-04-19 ltfinal.dtx v2.3b \@uclclist: Add \DeclareLowercaseMapping, \DeclareTitlecaseMapping and \DeclareUppercaseMapping	1075
General: Add dedicated rollback code to revert data structures (hook-args).	296	2023-04-19 lthooks.dtx v1.1c __hook_gput_next_do:nn: Initialise hook structure when adding 'next'	
Messages 'too-many-args', 'without-args' and 'one-time-args' added (hook-args).	288		
\AddToHookNextWithArguments: Add \AddToHookNextWithArguments (hook-args).	291		
\AddToHookWithArguments: Add \AddToHookWithArguments			

code (gh/1052).	277
\hook_if_empty:n: Simpler and faster version (gh/1052).	284
2023-05-12 ltxref.dtx v1.1q	
\label: (UFI)added a hook with argument	668
(UFI)extended to store five arguments	668
2023-05-13 ltmath.dtx v1.2n	
\leqno: add \ignorespaces gh/1059	705
2023-05-15 ltfiles.dtx v1.2t	
\IfFileExists@: Use expl3 file existence test	395
\IfFileExists@@: Macro added . . .	395
2023-05-21 ltcmdhooks.dtx v1.0h	
__hook_guess_arg_count:nw: Macro added (cmd-args).	318
__hook_make_prefixes:w: Changes to allow support arguments in cmd hooks (cmd-args).	310
2023-05-21 lthooks.dtx v1.1d	
__hook_if_cmd_hook:w: Changes to allow support arguments in cmd hooks (cmd-args).	286
__hook_make_usable:nn: Changes to allow support arguments in cmd hooks (cmd-args).	225
__hook_try_declaring_generic_hook:wn:	
Changes to allow support arguments in cmd hooks (cmd-args).	241
__hook_try_declaring_generic_next_hook:nn:	
Changes to allow support arguments in cmd hooks (cmd-args).	239
__hook_new:nn: Changes to allow support arguments in cmd hooks (cmd-args).	224
\c__hook_parameter_cmd./after_tt:	
Token lists added (cmd-args). . .	246
2023-05-26 ltcmd.dtx v1.1e	
__cmd_defaults_error:w: Use simpler variant \cs_generate_- from_arg_count:NNno	123
2023-05-30 ltfinal.dtx v2.3c	
\@uclist: Fix a typo in implementation of \DeclareLowercaseMapping, etc.	1075
2023-06-06 lthooks.dtx v1.1e	
__hook_code_gset_auxi:een:	
Short-circuit when the hook is declared without args (gh1078). .	248
2023-06-16 ltfiles.dtx v1.2u	
\IfFileExists@@: Support piped input	395

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	
\!	aa523, b431, b433
\!	<u>I214</u>
\"	v441, aa524, d508, d509, s226, s378, s419, s457, s468, s579, s611, s638, s646, s652, s656, s662, s666, s672, s678, s685, s686, s692, s696, s746, s792, s1244, s1262, s1268, s1272, s1278, s1282, s1288, s1294, s1301, s1302, s1308, s1312, s1314, s1421, E171, E198
\#	i446, v428, aa507, a65, a78, b6, b14, b531, f801
\\$	aa508, a77, b4, b13, f800, s307, s444, s451, s565, s804, s811, E631, E638
\\$	<u>1098</u>
\%	v430, aa509, a78, a108, a110, a130, b14, b529, f801, s491, s493, E640, E760, H142, U1460, U1461
\&	aa510, a77, b5, b13, b530, f800, U378
\'	v440, aa525, b551, s227, s379, s421, s455, s465, s581, s591, s597, s599, s602, s604, s612, s618, s624, s626, s629, s631, s639, s643, s650, s654, s659, s664, s667, s669, s676, s681, s682, s689, s694, s697, s747, s794, s813, s815, s816, s817, s820, s822, s823, s824, s826, s827, s1238, s1259, s1266, s1270, s1275, s1280, s1283, s1285, s1292, s1297, s1298, s1305, s1310, s1313, s1321, s1322, s1369, s1370, s1375, s1376, s1387, s1388, s1393, s1394, s1422, s1423, s1437, s1438, s1439, s1440, s1453, s1454, A775, B253, E161, H583, I254, K290, K311, L72, Y588
\(g2225, I345
\(<u>I271</u>
\)	b551, g2247, I346
\)	<u>I271</u>
*	v433, U1141, U1272, U1386, U1462
*	<u>I251</u>
\+	L72
\+	<u>1096</u>
\,	b432, b434, B506, H583, I7, I8, I40, I167, I169, I172, I197, I220, I221, I237
\,	<u>1137</u>
\-	v435, aa224, aa264, b378, f24, p417, p480, s416, s417, s574, s788, s789, H583, K289, K310, L72
\-	<u>f835</u> , <u>378</u>
\.	v434, b431, b433, r45, r115, r173, s228, s380, s452, s453, s474, s587, s588, s614, s615, s641, s748, s818, s825, s1243, s1325, s1326, s1335, s1336, s1345, s1346, s1363, s1424, s1425, s1461, s1462, E173, E199
\.	<u>1125</u>
\..default	<u>488</u>
.code	<u>V4</u>
.if	<u>V4</u>
.ifnot	<u>V4</u>
.store	<u>V4</u>
.usage	<u>V4</u>
\`	i202, i277, v382, v436, a100, f25, U377
\`	<u>1092</u>
\:	<u>I214</u> , I224, I225, I226, I242, <u>I252</u> , I252, f795, b432, b434
\;	b432, b434, B500, I198
\;	<u>I214</u> , <u>699</u>
\<	v431, s575, s739, H583, L71, L109
\=	s229, s381, s473, s749, s1241, s1315, s1316, s1331, s1332, s1354, s1355, s1356, s1381, s1382, s1407, s1408, s1441, s1442, s1443, s1444, s1459, s1460, s1467, s1468, A775, E179, K290, K311, L71
\>	v432, s572, s740, H583, I226, I241, I242, I252, L71
\?	aa525, b431, b433
\@ commands:	
\@_cmd_type_cases:NnnnTF	<u>1144</u>
\c @_hash_t1	<u>1150</u>
\c @_hashes_t1	<u>1150</u>
\@_if_file_hook:wTF	<u>1146</u>
\@_include_legacy_code_chunk:n	<u>1151</u>
\g @_input_file_seq	<u>1142</u>
\@_replacing_args_false:	<u>1151</u>
\@_set_curr_file:nNN	<u>W387</u>
\@_set_curr_file_assign:nnnNN	<u>W387</u>
\@_start_expandable:nNNNNn ..	<u>1150</u>
\@@\textvisiblespace_\meta_{\hook} }	<u>221</u>
\@@\next\textvisiblespace_\meta_{\hook}	<u>221</u>

File Key: a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\@_toplevel\textvisiblespace\meta{hook}	221	d272, d424, f231, f288, f449, f572, f590, f800, g404, g468, g499, g640, g652, g683, g1048, g1054, g2469, g2499, g2513, g2520, g2549, g2646, g2686, g2697, g2723, g2729, g2736, h1708, h1719, h2490, h2504, h2505, h2506, h2507, h2508, h2513, h2516, h2517, h2518, h2528, h2538, h2559, h2563, h2564, h2566, h2572, h2583, h2584, h2589, h2594, h2599, h2615, h2788, h2789, h2790, l277, n139, n146, n153, n154, n155, n156, p482, r746, r765, s560, B251, H365, H370, H375, H385, H389, H393, H417, I370, I523, K322, K481, K483, L73, L170, L180, L194, 360, M139
\@par	339	
\@TeXversion	6, 1	
@botlist commands:		
\@botlist:	Y1033, Y1127, Y1286	
@botnum commands:		
\@botnum:	Y1007	
\@car	82	
\@cdr	82	
@citere commands:		
\@citereb:	R36, R65, R82	
@colht commands:		
\@colht:	Y545, Y559	
@column commands:		
\@colnum:	Y1192, Y1360	
\@cons	82	\` p53, 1109
\@currdir	6, 1	\{ i444, v426, aa514, a6, a10, a77, b2, b13, g571, g1798, l22, s308, s562, B249, H416, I59, I167
@currname commands:		\} i445, v427, aa515, a11, a77, b3, b13, l21, s309, s563, B250, H416, I59
\@currname:	r753	
\@dblarg	81	\langle addto-cmd\rangle 194
@dbldeferlist commands:		\langle cmd\rangle\` 144
\@dbldeferlist:	Y1759, Y1801	\langle cmd\rangle\` (arg\`<num>) 146
@dbltopnum commands:		\langle cmd\rangle\` \` 144
\@dbltopnum:	Y1658, Y1785	\langle cmd\rangle\` code 144
@deferlist commands:		\langle cmd\rangle\` defaults 144
\@deferlist:	Y1118, Y1215, Y1276, Y1384, Y1428, Y1457, Y1515, Y1547, Y1633, Y1673	\langle filename\rangle 948
\@ifclasslater	877	\langle function\rangle 165
\@ifclassloaded	877	\` v422, v672, v708, v733, aa506, a77, a94, b13, b381, b437, b466, b467, b484, f800, h2564, l19, l20, l21, l22, l25, p422, B252, H413, H414, H519, H534, N36, N38, R37, U371
\@ifclasswith	877	\] v438, aa527, b551, I348
\@ifdefinable	81	\] I289, I486
\@ifnextchar	81	\` v423, v424, v429, aa255, aa256, aa257, aa258, aa259, aa260, aa261, aa262, aa263, aa512, aa518, aa519, aa520, aa521, aa555, aa556, aa557, aa558, aa559, aa560, aa561, aa562, aa563, a66, a75, a78, a122, a333, b7, b9, b11, b14, b380, b437, b438, b457, b458, b479, b480, f20, f801, g6, g7, g1737, g2093, g2098, g2900, p482, p484, p486, s231, s286, s382, s456, s466, s558, s644, s651, s655, s660, s665, s670, s677, s683, s684, s690, s695, s750, s1239, s1256, s1260, s1267, s1271, s1276, s1281, s1286, s1293, s1299, s1300, s1306, s1311, s1323, s1324, s1341, s1342, s1349, s1350, s1364, s1365, s1366, s1395,
\@ifpackagelater	877	
\@ifpackageloaded	877	
\@ifpackagewith	877	
\@ifstar	81	
\@ifundefined	81	
\@missingfileerror	878	
\@namedef	81	
\@nameuse	81	
\@restorepar	339	
\@setpar	339	
@topnum commands:		
\@topnum:	Y1053	
\[. v437, aa526, A43, A54, A76, A87, I347		
\[. I289, I462, I127		
\` i443, i586, v425, aa511, a48, a49, a77, a250, a251, a252, a253, a256, a263, a264, a265, a266, a269, a276, a277, a278, a279, a282, a289, a295, a296, a300, a302, a303, a307, a312, a313, a316, a322, b13, b377,		

File Key: a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=lttoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

s1396, s1417, s1418, s1419, s1420,
 E169, E197, H134, H143, U1142,
 U1143, U1144, U1225, U1228,
 U1231, U1273, U1274, U1275,
 U1357, U1360, U1363, U1387,
 U1388, U1389, U1447, U1450, U1453
 _ s314, B254, I269,
 I270, aa513, b8, f801, b14, a78, 1105
 \` v439, aa528, s232, s383, s420,
 s454, s464, s580, s642, s649, s653,
 s658, s663, s668, s675, s679, s680,
 s688, s693, s751, s793, s1237, s1258,
 s1265, s1269, s1274, s1279, s1284,
 s1291, s1295, s1296, s1304, s1309,
 A775, E175, H583, K290, K311, L72
 \| aa529, s561, t169, t180, B573, B574
 \~ aa516, a78, b10, b14, f801,
 l20, p423, s239, s287, s384, s467,
 s559, s645, s657, s661, s671, s687,
 s691, s752, s1240, s1257, s1261,
 s1273, s1277, s1287, s1303, s1307,
 s1351, s1352, s1353, s1405, s1406,
 E187, E207, H514, H529, H544, H587

A

\A aa252, aa531, aa552
 \a s223, L1, aa243, aa532, aa543, 1096
 \AA s240, s428, s526, b443, 1121
 \aa s245, s422, s536, b443, 1121
 \abovedisplayshortskip I531, b413
 \abovedisplayskip I524,
 I526, I528, I529, I530, I531, b412, 1106
 \accent .. s73, s393, s423, s479, s763, 1123
 \ActivateGenericHook
 h2639, h2641, h2646, h2648, 209

\active a122, H413, H414,
 H513, H519, H528, H534, H543,
 H585, I254, I269, U1142, U1143,
 U1144, U1225, U1228, U1231,
 U1273, U1274, U1275, U1357,
 U1360, U1363, U1387, U1388,
 U1389, U1447, U1450, U1453,
 Y588, a333, b10, b11, a67, b457,
 b458, b466, b467, b479, b480, b484

\acute B516
 add commands:
 add_to_callback d813
 \add_to_callback 46
 \addcontentsline O70, O80, O159, P16, 676
 \AddEverypageHook 963
 \Adding 1081
 \addpenalty p271, J124, J170,
 J175, O50, Y340, Y1157, Y1323, 372
 \AddThispageHook 963

\addtocontents
 O164, O171, O177, O180, 676
 \addtocounter 449
 \addtocounter t6, t18, 1093
 \AddToHook h2651, h2809,
 x152, H38, H39, H40, H310, H311,
 H312, H313, R72, U1043, U1044,
 U1045, W562, W564, W574, W575,
 W576, W577, X190, X473, X496, 299
 \AddToHookNext
 h2662, h2807, W563, X497, 201
 \AddToHookNextWithArguments . h2662, 197
 \AddToHookWithArguments h2651, 195
 \addtolength 457
 \addtolength u16, I526, I528
 \AddToNoCaseChangeList aa576, 1148
 \addtoversion y20, y139
 \addvspace p234, H343,
 J124, J171, J172, J176, J224, O50, 1092
 \adjdemerits b336
 \AE s241, s398,
 s527, s768, s1127, s1437, s1441, aa654
 \ae s246, s401,
 s537, s772, s1133, s1439, s1443, aa654
 \afterassignment s212, s220, v328,
 I199, f268, f274, f317, b501, b504, 93
 \AfterEndEnvironment H306, 214
 \aftergroup v91, v347, v358, v362,
 x202, x268, z114, z121, z129, z404,
 D64, H517, H532, H547, I445, I446,
 K148, Y608, Y609, Y666, Y667, 1102
 \AfterLastShipout W574
 \afterpreamble 384
 \aleph B308
 \allocationnumber n74, d52, d53,
 d54, d91, d213, L4, L9, X34, aa58,
 aa59, aa60, b37, b57, b69, b71,
 b143, b144, b145, b195, b196, b237,
 b238, b239, b252, b253, b254, b271,
 b277, b283, b284, b297, b298, b299, 352
 \allowbreak I40, f877, f878, f897, f899, b508
 \Alph 449
 \Alph t140, 961
 \alph 449
 \alpha t139
 \alpha B268
 \amalg B379
 \AmSfont 497
 \and 803
 \and O14, O27
 \angle B337
 \approx B422
 \arabic 449
 \arabic t79, t87, t136, N33, X350, 961

File Key: a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=lttoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\arccos	I13	B
\arcsin	I10	\b s233, s389, s475, s759, s1248, 1108
\arctan	I16	\backslash B251, B590
\arg	I26	\bar B520
\ArgumentSpecification		\baselineskip x186, x187, x188, x190, x191, B510, I171, I172, I191, I197, I201, K299, K318, L198, M136, M314, M373, X222, X273, Y244, Y275, Y626, Y641, Y685, Y700, b409, b436, b499, b535, 1117
array (env.)	L168	\baselinestretch v319, x120, x121, x166, x167, x184, x245, 1079
\array	L168	\baselinestretch 1099
\arraycolsep		\batchmode r691, r722, r723, y106, e89, e95, e105, aa672, aa693, 400
... I373, I374, I536, I537, L258, L338		\BCPdata aa565, aa588, aa592
\arrayrulewidth	L324, L338, L346, L347, L359, L363, L366, L376, L378	\BeforeBeginEnvironment H306, 214
\arraystretch	L186, L187, L342	\BeforeClearDocument W578
\Arrowvert	B569	\begin h2688, i66, i504, l246, l248, r345, r397, x7, B4, B102, C4, H174, H175, I466, I478, M461, O14, O17, U681, V209, X394, Z3, aa489, aa493, f709, g1306, g1416, 215
\arrowvert	B567	\begingroup 1079
\asciispace		\belowdisplayshortskip I530, b415
H473, H475, H478, H481, H482, H501		\belowdisplayskip I529, b414
\ast	B232, B395	\beta B269
\asymp	B449	\bezier 767
\AtBeginDocument		\bezier M682, M683, M805, M806, M821, M823
... r126, r181, w737, R54, U1034, 212		\bf 1087
\atbegindocumenthook	384	\bfdefault
\AtBeginDvi	X412, X448, Y86, 979	... A15, A261, A267, A268, A269, A270, A310, A311, A312, A313, A322, A358, A382, A401, A442, A476, B92, B104, B106, B114, 579
\AtBeginEnvironment	H306, 214	\bfseries A13, A14, A253, A254, A302, A307, A308, A349, A352, A353, A378, A380, A381, A474, A475, D19, G17, G34, G51, N36, N38, R40, X395, 571
\AtBeginShipout	X453, X496, 1146	bfseries A420
\AtBeginShipoutAddToBox	X501, 962	bfseries/defaults A420
\AtBeginShipoutAddToBoxForeground		\bgroup b450, 1094
... X501, 962		\bibcite R7, R9, R10, 1113
\AtBeginShipoutBox	X494, 962	\bibdata R45, R49
\AtBeginShipoutDiscard	X500, 963	\bibitem R3
\AtBeginShipoutFirst	X456, X498, 963	\bibliography 842
\AtBeginShipoutInit	X495, 963	\bibliography R47
\AtBeginShipoutNext	X454, X496, 963	\bibliographystyle 842
\AtBeginShipoutOriginalShipout	X509, 962	\bibliographystyle R52
\AtBeginShipoutUpperLeft	X501, 962	\bibstyle R45, R57
\AtBeginShipoutUpperLeftForeground		\Big B623, B626, B635, B637, I44, I45, I46
... X501, 962		\big B624, B636, I41
\AtEndAfterFileList	W576	
\AtEndDocument	H76, U1034, W580, 213	
\AtEndDvi	X463, X468, 959	
\AtEndEnvironment	H306, 214	
\AtEndOfClass	I461, U1034, 935	
\AtEndOfPackage	U543, U562, U660, U671, U1034, V52, 935	
\AtEndPreamble	215	
\AtNextShipout	963	
\atopwithdelims	I57, I58, I59	
\attribute	d79, 43	
\attributedef	d79, d223	
\attributezero	d223	
\AtVeryEndDocument	W575	
\AtVeryVeryEnd	W577	
\author	803	
\author	O8, O24, O32	

File Key: a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

```

\bigbreak ..... f879, f900, b515
\bigcap ..... B345
\bigcirc ..... B392
\bigcup ..... B346
\Bigg ..... B630, B639, I50, I51, I52
\bigg ..... B628, B638, I47, I48, I49
\Biggl ..... I50
\biggl ..... I47
\Biggm ..... I51
\biggm ..... I48
\Biggr ..... I52
\biggr ..... I49
\Bigl ..... I44
\bigl ..... I41
\Bigm ..... I45
\bigm ..... I42
\bigodot ..... B353
\bigoplus ..... B352
\bigotimes ..... B351
\Bigr ..... I46
\bigr ..... I43
\bigskip ..... p401, b520
\bigskipamount ..... p403, p404, P391, b519
\bigsqcup ..... B356
\bigtriangledown ..... B361, B362
\bigtriangleup ..... B360, B363
\biguplus ..... B344
\bigvee ..... B342
\bigwedge ..... B343
\binoppenalty ..... b323
\bmod ..... I35
\boldmath ..... q14, A613
bool commands:
    \bool_gset_false:N ..... S226, X15, X81, X90, h15
    \bool_gset_true:N ..... S221, X10, X120, X344, h10
    \bool_if:NTF ..... i162, i166, i178, i237, i241, i253, i380, S232, V172, X21, X88, X365, g100, g111, g123, g127, g136, g138, g148, g157, g158, g161, g166, g168, g254, g399, g408, g410, g465, g496, g511, g560, g593, g637, g647, g680, g688, g693, g694, g702, g725, g738, g816, g862, g863, g922, g935, g952, g972, g1834, g2068, g2461, g2472, h21, h1842, h1851, h1937, h1946, 189
    \bool_lazy_and:nnTF ..... h2359, h2830, X66, X112, X376, g676, g864, h275, h1844, h1939
    \bool_lazy_and_p:nn ..... h2362
    \bool_lazy_any:nTF ..... g419
    \bool_lazy_or:nnTF ..... g73, g2223, g2245, h856
    \bool_new:N ..... S217, V24, X6, X188, X203, g16, g18, g20, g31, g32, g34, g35, g37, g40, g43, g44, g45, h6, h24
    \bool_set_false:N ..... V51, g57, g83, g192, g235, g385, g386, g387, g388, g389, g390, g411, g473, g503, g528, g542, g564, g565, g566, g581, g643, g686, g697, g698, g714, g715, g716, g718, g722, g729, g871, g872, g873, g1724, g2269, h1834, h1929
    \bool_set_true:N ..... V48, g62, g193, g224, g384, g409, g480, g487, g488, g502, g689, g690, g745, g746, g752, g753, g759, g760, g767, g768, g769, g889, g907, g1729, g2275, h1831, h1926
    \bool_while_do:nn ..... h1533, h1614
    \c_false_bool ..... i141, i151, i370, g2069, g2433, g2738, g2843, 307
    \c_true_bool ..... i146, g2070, g2435, g2739, g2841, 189
bool internal commands:
    \g__mark_debug_bool ..... S217, S221, S226, S232
\BooleanFalse ..... g1717, g2062, g2738
\BooleanTrue ..... g1716, g2061, g2738
\bordermatrix ..... I185
\bot ..... B321
\botfigrule ..... Y746, Y2363
\botmark ..... T79, Y651, Y710, 847
\bottomfraction ..... P275, Y2332
\bowtie ..... B482
\Box ..... A726
\box ..... 344
box commands:
    \box_dp:N ..... X194
    \box_gclear:N ..... n82
    \box_gset_to_last:N ..... n16, n42, n110
    \box_ht:N ..... X193, X302, 967
    \box_if_empty:NTF ..... X78, X97
    \box_if_horizontal:NTF ..... X234, X286
    \box_if_vertical:NTF ..... S60, S284, S315, X208, X259
    \box_move_up:nn ..... X248, X302
    \box_new:N ..... n77, S43, X23, X25, X187, X204
    \box_set_dp:Nn ..... X221, X230, X247, X272, X283, X301, X332
    \box_set_eq:NN ..... X149, X179, X184
    \box_set_eq_drop:NN ..... X93
    \box_set_ht:Nn ..... X220, X229, X246, X271, X282, X300, X331
    \box_set_to_last:N ..... S57

```

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfnctcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\box_set_wd:Nn	X219, X245, X270, X299	\capitaldieresis
\box_use:N	X126, X225, X250, X278, X303, X333	. s842, E170, E171, E198, E695, E946
\box_use_drop:N	n79	\capitaldotaccent
\box_wd:N	X195, X296, X304	. s848, E172, E173, E199, E696, E952
box internal commands:		\capitalgrave
\l_mark_box	S43,	. s838, E174, E175, E200, E697, E942
S53, S57, S60, S62, S64, S69, S76, 857		\capitalhungarumlaut
\boxmaxdepth	M475,	. s843, E176, E177, E201, E698, E947
M503, M533, M611, M628, Y490,		\capitalmacron
Y510, Y550, Y719, Y728, Y768, b395		. s847, E178, E179, E202, E699, E951
\brace	I59	\capitalnewtie
\braceleft	B553, B557, B558, B560, B562	. s852, E190,
\braceright	B555, B559, B561	E191, E203, E700, E1017, E1018, 637
\bracerule	B554, B559, B561	
\bracevert	B556, B558, B562	\capitalogonek
\brack	I58	. s835, E180, E181, E204, E701, E941
\break	p116, f880, f901, b508, b513, 1113	\capitalring
\breve	B521	. s844, E182, E183, E205, E702, E948
\brokenpenalty	v691, b328	\capitaltie
\buildrel	B469, I162	. s850,
\bullet	B381	E184, E185, E206, E703, E1013, E1014
C		
\c	s234, s335, s337, s339, s341, s343, s345, s347, s349, s351, s372, s374, s392, s459, s478, s606, s608, s633, s635, s648, s674, s701, s704, s705, s706, s707, s708, s709, s710, s711, s712, s762, s1250, s1264, s1290, s1347, s1348, s1367, s1368, s1371, s1372, s1377, s1378, s1389, s1390, s1397, s1398, s1401, s1402, E167, E196, 1116	\capitaltilde
\cal	A776	. s841, E186, E187, E207, E704, E945
call commands:		\caption
\call_callback	d794	P4
\call_callback	47	\cases
callback commands:		I166, I167, I177, I179
\callback_descriptions	d979	\CaseSwitch
\callback_register	d682	aa576, 1148
\callback_descriptions	47	\catcode
\cap	B372	677
\capitalacute		\catcodetable
. s839, E160, E161, E193, E690, E943		d89, d109, 43
\capitalbreve		\catcodetable@atletter
. s846, E162, E163, E194, E691, E950		44
\capitalcaron		\catcodetable@initex
. s845, E164, E165, E195, E692, E949		44
\capitalcedilla		\catcodetable@latex
. s832, E166, E167, E196, E693, E940		44
\capitalcircumflex		\catcodetable@string
. s840, E168, E169, E197, E694, E944		44
char commands:		
\char_generate:nn	e142, g1546, g1842	K498
\char_set_catcode_active:N	... g2093	

File Key: a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

```

\char_set_catcode_active:n . . . . . g2332
\char_set_catcode_escape:N . . . . . i443
\char_set_catcode_group_begin:N i444
\char_set_catcode_group_end:N . i445
\char_set_catcode_other:N . . . . . g1832
\char_set_catcode_other:n . . . . . g1835
\char_set_catcode_parameter:N . i446
\char_set_catcode_parameter:n . g1836
\char_set_lccode:nn . . . . . g2098, g2342
\char_value_catcode:n . . . . . i375
\chardef . . . . . j2, r52, r122,
    s18, d22, d26, d38, d47, d48, d89,
    v14, d157, d224, L4, L9, U1148,
    U1279, U1396, aa42, aa44, aa48,
    aa67, aa171, aa172, aa173, aa174,
    aa175, aa176, aa177, 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,
    a67, a73, a74, b529, b530, b531, 1096
\charsubdef . . . . . aa353
\charzero . . . . . d224
\check . . . . . B522
\CheckCommand . . . . . f187, 1100
\CheckEncodingSubset E16, E61, E109,
    E110, E111, E156, E158, E352,
    E621, E832, E882, E938, E939,
    E1007, E1124, E1127, E1141, 631
\chi . . . . . B288
\choose . . . . . I57
\circ . . . . . B391
\circle . . . M450, M605, M807, M824, 1083
\citation . . . . . R11, R39, R67, R84
\cite . . . . . 842
\cite . . . . . R12, 843
\clap . . . . . K502, 1137
class//after . . . . . 934
class//before . . . . . 934
class/after . . . . . 934
class/before . . . . . 934
\ClassError . . . . . l84
\ClassInfo . . . . . l84
\ClassNote . . . . . l136, 1144
\ClassNoteNoLine . . . . . l136
\ClassWarning . . . . . l84
\ClassWarningNoLine . . . . . l84
\cleaders . . . . . b549, B548, B551
\cleardoublepage . . . . . Y140
\ClearHookNext . . . . . h2673, 197
\ClearHookRule . . . . . h2720, h2823, 201
\clearpage . . . . . .
    r297, r324, r328, r359, r382, r385,
    r406, r424, H17, H79, H172, Y127,
    Y140, Y145, Y202, Y409, Y412,
    Y416, Y457, Y463, Y2209, Y2226, 936
\cline . . . . . L367
clist commands:
\clist_clear:N . . . . . V32
\clist_gclear:N . . . . . h1532, h1613
\clist_gput_left:Nn . . . . h1443, h1475
\clist_gput_right:Nn . . . . h1445, h1477
\clist_if_empty:NTF . . . . h1861, h1956
\clist_if_exist:NTF . . . . h121
\clist_if_in:NnTF . . . . V98
\clist_map_inline:Nn V81, V104, V160
\clist_map_inline:nn V195, h887, h896
\clist_map_variable:Nnn . . . . V49
\clist_new:N . . . . . V23, h123, h140
\clist_put_right:Nn . . . . .
    . . . . . V22, V77, V94, V99, V115, V125
\clist_remove_all:Nn . . . . . V95, V116
\clist_use:Nn . . . . . h1863, h1958
\clubpenalty . . . . . r8, r25,
    r95, r152, v689, J128, J194, J196,
    O100, O106, O130, O135, b325, 351
\clubsuit . . . . . B331
cmd internal commands:
\__cmd_add_arg:n . . . . . g1482,
    g1493, g1498, g1499, g1533, g1621,
    g1628, g1703, g1716, g1717, g1791,
    g1848, g1855, g1868, g1868, g1873, 153
\__cmd_add_arg_spec:n . . . . .
    . . . . . g526, g540, g600, g674, g674, g709
\__cmd_add_arg_spec_mandatory:n . . . . .
    . . . . . g572, g582, g588, g674, g700
\__cmd_add_default: . . . . .
    . . . . . g777, g815, g832,
    g883, g886, g893, g903, g970, g979, 135
\__cmd_add_default:n . . . . . g784,
    g824, g840, g883, g883, g900, g914, 135
\__cmd_add_default_E:nn . . . . .
    . . . . . g792, g883, g898, g948
\__cmd_add_expandable_grabber:nn . . . . .
    . . . . . g927,
    g940, g951, g971, g981, g988, g988
\__cmd_add_expandable_type_+:w g905
\__cmd_add_expandable_type_D:w . . . . .
    . . . . . g910, g910
\__cmd_add_expandable_type_D_- aux:NN . . . . . g910, g916, g933
\__cmd_add_expandable_type_D_- aux:NNN . . . . . g910, g917, g920
\__cmd_add_expandable_type_D_- aux:NNNn . . . . . g910, g911, g912, g976
\__cmd_add_expandable_type_E:w . . . . .
    . . . . . g946, g946

```

File Key: a=ltdirchk.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=ltcntr.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

```

\__cmd_add_expandable_type_E_-
  aux:n ..... g946, g950, g962
\__cmd_add_expandable_type_m:w ..
  ..... g968, g968
\__cmd_add_expandable_type_R:w ..
  ..... g975, g975
\__cmd_add_expandable_type_t:w ..
  ..... g977, g977
\__cmd_add_grabber:N .....
  ..... g778, g785, g798,
  g817, g825, g833, g841, g855, g855, 137
\__cmd_add_type_!:_w .....
  ..... g749
\__cmd_add_type_+_w .....
  ..... g742
\__cmd_add_type_=: .....
  ..... g764
\__cmd_add_type_>:_w .....
  ..... g756
\__cmd_add_type_b:w .....
  ..... g774, g774
\__cmd_add_type_D:w .....
  ..... g781, g781
\__cmd_add_type_E:w .....
  ..... g789, g789
\__cmd_add_type_m:w .....
  ..... g813, g813
\__cmd_add_type_R:w .....
  ..... g821, g821
\__cmd_add_type_t:w .....
  ..... g829, g829
\__cmd_add_type_v:w .....
  ..... g837, g837
\__cmd_allowed_token_check:N ...
  .. g524, g536, g557, g578, g616, g616
\l__cmd_arg_spec_t1 .....
  ..... g11,
  g92, g383, g471, g504, g558, g691, 130
\__cmd_arg_to_keyvalue:nnn .....
  ..... g771, g2152, g2152
\__cmd_arg_to_keyvalue_auxi:nnn .....
  ..... g2152, g2167, g2169
\__cmd_arg_to_keyvalue_auxii:Nnnn .....
  ..... g2152, g2172, g2175
\__cmd_arg_to_keyvalue_auxiii:nnn .....
  ..... g2152, g2178, g2181
\__cmd_arg_to_keyvalue_auxiv:Nnnn .....
  ..... g2152, g2184, g2187
\__cmd_arg_to_keyvalue_auxv:nn ..
  ..... g2152,
  g2173, g2179, g2185, g2191, g2193
\__cmd_arg_to_keyvalue_braces:nnn .....
  ..... g2152, g2154, g2157
\__cmd_arg_to_keyvalue_loop:w .....
  ..... g2152, g2195,
  g2198, g2210, g2212, g2227, g2248
\__cmd_arg_to_keyvalue_loop_-
  group:n .....
  ..... g2152, g2204, g2209
\__cmd_arg_to_keyvalue_loop_N_-
  type:N .....
  ..... g2152, g2201, g2213
\__cmd_arg_to_keyvalue_loop_-
  space:w .....
  ..... g2152, g2205, g2211
\__cmd_arg_to_keyvalue_math:w ...
  ..... g2152,
  g2226, g2230, g2249, g2252, g2254
\__cmd_arg_to_keyvalue_math_-
  group:n .....
  ..... g2152, g2236, g2251
\__cmd_arg_to_keyvalue_math_-
  type:N .....
  ..... g2152, g2233, g2241
\__cmd_arg_to_keyvalue_math_-
  space:w .....
  ..... g2152, g2237, g2253
\__cmd_arg_to_keyvalue_set_-
  default:nn .....
  ..... g2152, g2216, g2244, g2255, 174
\__cmd_arg_to_keyvalue_set_-
  keyvalue:mn g2152, g2220, g2257, 174
\l__cmd_args_i_t1 .....
  ..... g13, g263, g269, g274, g275, g277, 113
\l__cmd_args_ii_t1 .....
  ..... g14,
  g273, g275, g277, g315, g320, g327, 113
\__cmd_args_process: .....
  ..... g253, g313, g313, 135
\__cmd_args_process_aux:n .....
  ..... g313, g326, g330
\__cmd_args_process_loop:nn .....
  ..... g313, g319, g322
\l__cmd_args_t1 .....
  ..... g12, g216,
  g241, g258, g263, g269, g288, g317,
  g320, g333, g334, g1418, g1419,
  g1424, g1427, g1599, g1633, g1870, 123
\__cmd_bad_arg_spec:wn g441, g446,
  g455, g464, g510, g523, g533, g534,
  g539, g550, g556, g577, g668, g668
\__cmd_bad_def:wn .....
  ..... g397, g405, g434, g469,
  g500, g515, g597, g605, g613, g632,
  g641, g653, g668, g673, g684, g706, 143
\__cmd_bool_reverse:N .....
  ..... g2066, g2066, g2874
\__cmd_break_point:n ... g95, g97,
  g97, g668, g673, g1037, g1044, g1242
\__cmd_cant_copy:nwn .....
  ..... g1034,
  g1044, g1044, g1148, g1211, g1240
\__cmd_check_definable:nNTF .....
  ..... g2329, g2329,
  g2742, g2755, g2768, g2773, g2798,
  g2811, g2824, g2832, g2881, g2887
\__cmd_check_definable_aux:nN .....
  ..... g2329, g2330, g2333, 179
\__cmd_check_end:n g1207, g1209, g1215
\__cmd_check_end:Nn .....
  .. g1194, g1195, g1207, g1207, g1315
\__cmd_check_end:w g1207, g1217, g1220
\__cmd_chk_if_free_cs:N .....
  ..... g2301, g2891, g2892

```

File Key: a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntr1.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspc.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfnctcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

```

\__cmd_cmd_if_xparse:NTF .... 1143
\__cmd_cmd_if_xparse_aux:N ... g2396
\__cmd_cmd_type_cases:Nnnnn . g2396
\__cmd_cmd_type_cases:NnnnnTF ...
    . . . . . g1029, g1235, g2396, g2418, 143
\__cmd_copy:NN .... g1022, g1024,
    g1024, g1058, g1073, g1107, g1225, 144
\__cmd_copy_command:nnNN ....
    . . . . . g1030, g1059, g1059, 144
\__cmd_copy_command:NnNNnnnn ...
    . . . . . g1059, g1064, g1066, 144
\__cmd_copy_environment:nnNN ...
    . . . . . g1032, g1181, g1181
\__cmd_copy_environment:Nnnnnnn .
    . . . . . g1181, g1187, g1190
\__cmd_copy_environment_end:nnNN
    . . . . . g1033, g1192, g1192
\__cmd_copy_environment_end_-
    aux:nnNN .... g1192, g1196, g1199
\__cmd_copy_expandable:nnN ....
    . . . . . g1126, g1131,
    g1134, g1161, g1171, g1177, g1179
\__cmd_copy_expandable:nnNN ....
    . . . . . g1031, g1072,
    g1076, g1078, g1091, g1093, g1104
\__cmd_copy_expandable:NnNNNnnn
    . . . . . g1072, g1088, g1101, g1109, 144
\__cmd_copy_expandable_signature:NnNNNNnnn
    . . . . . g1084, g1099, g1126, g1126, 144
\__cmd_copy_grabber_{type}:w ... 146
\__cmd_copy_grabber_D:w .....
    . . . . . g1151, g1151, g1164, g1165
\__cmd_copy_grabber_D_alt:w ...
    . . . . . g1151, g1163, g1166
\__cmd_copy_grabber_E:w .....
    . . . . . g1151, g1167, g1173
\__cmd_copy_grabber_E_long:w ...
    . . . . . g1151, g1173
\__cmd_copy_grabber_m:w .....
    . . . . . g1151, g1179, g1180
\__cmd_copy_grabber_m_long:w ...
    . . . . . g1151, g1180
\__cmd_copy_grabber_R:w g1151, g1165
\__cmd_copy_grabber_R_alt:w ...
    . . . . . g1151, g1166
\__cmd_copy_grabber_t:w g1151, g1174
\__cmd_copy_parse_grabber:w ...
    . . . . . g1126, g1138, g1142, 146
\__cmd_current_arg_int .....
    . . . . . g15, g25, g108, g137, g144, g206,
    g287, g291, g296, g297, g381, g393,
    g416, g472, g518, g543, g713, g888,
    g895, g1129, g1137, g1155, g1159,
    g1160, g1170, g1324, g1390, g1399, 129
\__cmd_declare_cmd:Nnn .....
    . . . . . g55, g55, g2750, g2758, g2769, g2774
\__cmd_declare_cmd_aux:Nnn .....
    . . . . . g55, g58, g63, g65
\__cmd_declare_cmd_code:Nnn .....
    . . . . . g93, g98, g98
\__cmd_declare_cmd_code_aux:Nnn .
    . . . . . g98, g102, g104
\__cmd_declare_cmd_code_expandable:Nnn
    . . . . . g98, g101, g132
\__cmd_declare_cmd_internal:Nnnm
    . . . . . g55, g84, g86, g199, 120
\__cmd_declare_env:nnnm .....
    . . . . . g178, g178, g2783, g2789, g2793, g2795
\__cmd_declare_env_internal:nnnm
    . . . . . g178, g194, g197
\__cmd_declare_expandable_-
    cmd:Nnn .....
    . . . . . g55, g60, g2806, g2814, g2827, g2833
\__cmd_defaults: .... g252, g260, g260
\__cmd_defaults_aux: .....
    . . . . . g260, g264, g265, g266, g271
\__cmd_defaults_bool .....
    . . . . . g16, g123, g138, g161, g718, g889, 114
\__cmd_defaults_def: g260, g262, g284
\__cmd_defaults_def:nn .....
    . . . . . g260, g289, g294
\__cmd_defaults_def:nnn .....
    . . . . . g260, g297, g299
\__cmd_defaults_error:w .....
    . . . . . g260, g267, g279
\__cmd_defaults_t1 g17, g124, g145,
    g244, g252, g288, g719, g890, g896, 114
\__cmd_delimiter_check:nnm .....
    . . . . . g571, g580, g657, g657
\__cmd_end_expandable:NNw .....
    . . . . . g340, g342, g342
\__cmd_end_expandable_aux:nNNNN .
    . . . . . g342, g345, g346
\__cmd_end_expandable_aux:w .....
    . . . . . g342, g343, g344
\__cmd_end_expandable_defaults:nnnNNn
    . . . . . g342, g349, g358, g367, g377, g378, 124
\__cmd_end_expandable_defaults:nnw
    . . . . . g342, g366, g370
\__cmd_end_expandable_defaults:nw
    . . . . . g342, g373, g374, g376
\__cmd_environment_bool ... g18,
    g83, g111, g193, g224, g235, g254,
    g410, g593, g2269, g2275, g2472, 114
\__cmd_environment_or_command: ..
    . . . . . g282, g396,

```

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=ltcntr.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

```

g428, g432, g514, g596, g603, g612,
g627, g671, g705, g1701, g1845,
g1852, g2285, g2289, g2470, g2470
\l__cmd_environment_str . . . .
    . . . . . g19, g114, g180, g181, g182,
    g183, g186, g190, g195, g223, g226,
    g227, g255, g2276, g2278, g2473, 114
\l__cmd_expandable_aux_name_t1 . .
    g21, g22, g925, g929, g938, g942, 114
\l__cmd_expandable_bool . g20, g57,
    g62, g136, g148, g192, g399, g408,
    g465, g496, g637, g647, g680, g725, 114
\l__cmd_expandable_grab_D:nnNNNwNN
    . . . . . g1874, g1898, g1904
\l__cmd_expandable_grab_D:NNNwNNn
    . . . . . g1874, g1875, g1878
\l__cmd_expandable_grab_D:NNNwNNnnn
    g1874, g1889, g1896, g1922, g2017, 167
\l__cmd_expandable_grab_D:Nw . .
    . . . . . g1874, g1900, g1903
\l__cmd_expandable_grab_D:w . .
    . . . . . g1874, g1874
\l__cmd_expandable_grab_D-
    alt:NNwn . . . . g1941, g1948, g2044
\l__cmd_expandable_grab_D-
    alt:NNNwNn . . . g1926, g1927, g1930
\l__cmd_expandable_grab_D_alt:Nwn
    . . . . . g1926
\l__cmd_expandable_grab_D_alt:w . .
    . . . . . g1926, g1926
\l__cmd_expandable_grab_E:w . .
    . . . . . g1961, g1961
\l__cmd_expandable_grab_E_aux:w . .
    . . . . . g1961, g1962, g1964, g1965, g1993
\l__cmd_expandable_grab_E_end:nnw
    . . . . . g1961, g1977, g1994
\l__cmd_expandable_grab_E-
    find:nnw . . . . g1961, g1991, g1992
\l__cmd_expandable_grab_E_find:w . .
    . . . . . g1961, g1982, g1990
\l__cmd_expandable_grab_E_long:w . .
    . . . . . g1961, g1963
\l__cmd_expandable_grab_E-
    loop:nnnNNw . . . .
        . . . . . g1961, g1969, g1973, g1984
\l__cmd_expandable_grab_E-
    test:nnw . . . . g1961, g1966, g1967
\l__cmd_expandable_grab_m:w . .
    . . . . . g1996, g1996, 147
\l__cmd_expandable_grab_m_aux:wNn
    . . . . . g1996, g1997, g1999, g2000
\l__cmd_expandable_grab_m_long:w . .
    . . . . . g1996, g1998
\l__cmd_expandable_grab_R:w . . .
    . . . . . g2002, g2002
\l__cmd_expandable_grab_R_alt:w . .
    . . . . . g2029, g2029
\l__cmd_expandable_grab_R_alt-
    aux:NNwNNn . . . g2029, g2030, g2033
\l__cmd_expandable_grab_R-
    aux:NNNwNNn . . . g2002, g2003, g2006
\l__cmd_expandable_grab_t:w . .
    . . . . . g2056, g2056
\l__cmd_expandable_grab_t-
    aux:NNwn . . . g2056, g2057, g2058
\l__cmd_flush_m_args: . . . . . g725,
    g744, g751, g758, g766, g776, g783,
    g791, g823, g831, g839, g844, g844, 135
\l__cmd_fn_code_t1 g29, g243, g258, 114
\l__cmd_fn_t1 . . . . . g28, g89, g242,
    g1420, g1487, g1511, g1517, g1527,
    g1537, g1546, g1551, g1555, g1586,
    g1612, g1620, g1622, g1627, g1629,
    g1640, g1641, g1646, g1647, g1652,
    g1653, g1658, g1659, g1664, g1665,
    g1670, g1671, g1676, g1677, g1682,
    g1683, g1713, g1719, g2270, g2477, 114
\l__cmd_function_t1 g24, g30, g88,
    g90, g107, g118, g119, g135, g143,
    g153, g156, g160, g162, g170, g176,
    g404, g468, g499, g640, g652, g683, 115
\l__cmd_get_arg_spec:N . . . .
    . . . . . g2304, g2304, g2882
\l__cmd_get_arg_spec:n . . . .
    . . . . . g2304, g2309, g2884
\l__cmd_get_arg_spec:NTF . . . . . g2292,
    g2292, g2306, g2311, g2318, g2324
\l__cmd_get_arg_spec_error:N . . .
    . . . . . g2267, g2267, g2307, g2320
\l__cmd_get_arg_spec_error:n . . .
    . . . . . g2267, g2273, g2314, g2327
\l__cmd_get_arg_spec_error_aux:n . .
    . . . . . g2267, g2271, g2277, g2280
\l__cmd_get_grabber>NN . . . .
    . . . . . g964, g980, g993, g993, 141
\l__cmd_get_grabber_auxi>NN . . .
    . . . . . g993, g996, g999, g1007
\l__cmd_get_grabber_auxii>NN . . .
    . . . . . g993, g1014, g1017
\l__cmd_grab_b:w . . . . . g1406, g1406
\l__cmd_grab_b_aux>NNw . . . . . g1406,
    g1407, g1409, g1411, g1413, g1414
\l__cmd_grab_b_end:Nw . . . .
    . . . . . g1406, g1417, g1422
\l__cmd_grab_b_long:w . . . . . g1406, g1408

```

File Key: a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

```

\__cmd_grab_b_long_obey_spaces:w . . . . . g1406, g1412
\__cmd_grab_b_obey_spaces:w . . . . . g1406, g1410
\__cmd_grab_D:w . . . . . g1433, g1433
\__cmd_grab_D_aux:NNnNN . . . . .
... g1416, g1477, g1479, g1484, g1696
\__cmd_grab_D_aux:NNnNNN . . . . .
g1435, g1440, g1445, g1450, g1456,
g1462, g1468, g1474, g1477, g1477
\__cmd_grab_D_call:Nw . . . . .
... g1481, g1541, g1541, g1698, 154
\__cmd_grab_D_long:w . . . . . g1433, g1438
\__cmd_grab_D_long_no_strip:w . . .
... g1433, g1459
\__cmd_grab_D_long_obey_spaces:w . . .
... g1433, g1448
\__cmd_grab_D_long_obey_spaces_-_
no_strip:w . . . . . g1433, g1471
\__cmd_grab_D_nested:NNnN . . .
... g1490, g1504, g1507
\__cmd_grab_D_nested:w . . .
... g1504, g1522, g1539
\__cmd_grab_D_no_strip:w g1433, g1453
\__cmd_grab_D_obey_spaces:w . . .
... g1433, g1443
\__cmd_grab_D_obey_spaces_no_-
strip:w . . . . . g1433, g1465
\__cmd_grab_E:nnNN . . . . . g1560,
g1562, g1568, g1574, g1580, g1584
\__cmd_grab_E:w . . . . . g1560, g1560
\__cmd_grab_E_finalise: . . .
... g1560, g1593, g1609, g1616
\__cmd_grab_E_long:w . . . . . g1560, g1566
\__cmd_grab_E_long_obey_spaces:w . . .
... g1560, g1578
\__cmd_grab_E_loop:NnN . . .
... g1560, g1589, g1604, g1606, g1613
\__cmd_grab_E_obey_spaces:w . . .
... g1560, g1572
\l__cmd_grab_expandably_bool . . .
... g31, g100,
g384, g409, g411, g473, g503, g528,
g542, g564, g581, g643, g686, g738, 126
\__cmd_grab_m:w . . . . . g1617, g1617
\__cmd_grab_m_1:w . . . . . g1631
\__cmd_grab_m_2:w . . . . . g1631
\__cmd_grab_m_3:w . . . . . g1631
\__cmd_grab_m_4:w . . . . . g1631
\__cmd_grab_m_5:w . . . . . g1631
\__cmd_grab_m_6:w . . . . . g1631
\__cmd_grab_m_7:w . . . . . g1631
\__cmd_grab_m_8:w . . . . . g1631
\__cmd_grab_m_9:w . . . . . g1631
\__cmd_grab_m_aux:NNnnnnnn . . .
g1631, g1631, g1640, g1646, g1652,
g1658, g1664, g1670, g1676, g1682
\__cmd_grab_m_long:w . . . . . g1617, g1624
\__cmd_grab_R:w . . . . . g1690, g1690
\__cmd_grab_R_aux:NNnN . . .
... g1690, g1691, g1693, g1694
\__cmd_grab_R_long:w . . . . . g1690, g1692
\__cmd_grab_t:w . . . . . g1706, g1706
\__cmd_grab_t_aux:NNw . . .
... g1706, g1707, g1709, g1710
\__cmd_grab_t_obey_spaces:w . . .
... g1706, g1708
\__cmd_grab_v:w . . . . . g1722, g1722
\__cmd_grab_v_aux:w . . .
... g1722, g1725, g1730, g1732, 163
\__cmd_grab_v_aux_abort:n . . .
... g1749, g1767, g1773,
g1786, g1805, g1828, g1830, g1838, 162
\__cmd_grab_v_aux_catcodes: . . .
... g1764, g1796, g1830, g1830, 161
\__cmd_grab_v_aux_loop:N . . .
... g1759, g1765, g1769, g1783
\__cmd_grab_v_aux_loop>NN . . .
... g1759, g1772, g1775
\__cmd_grab_v_aux_loop_end: . . .
... g1759, g1780, g1788, g1819
\__cmd_grab_v_aux_put:N . . .
... g1763, g1782,
g1798, g1816, g1824, g1858, g1858
\__cmd_grab_v_aux_test:N . . .
... g1748, g1759, g1759
\__cmd_grab_v_bgroupt: . . .
... g1744, g1794, g1794
\__cmd_grab_v_bgroupt_loop: . . .
... g1794, g1799, g1801, g1817, g1825
\__cmd_grab_v_bgroupt_loop:N . . .
... g1794, g1804, g1807
\__cmd_grab_v_group_end: . . .
... g1722, g1753, g1790, g1840, 161
\__cmd_grab_v_long:w . . . . . g1722, g1727
\__cmd_grab_v_token_if_char:NTF .
g1761, g1777, g1809, g1866, g1866, 162
\g__cmd_grabber_int . . .
... g27, g1006, g1010, 114
\__cmd_if_recursion_tail_stop_-
do:Nn . . . . . g46, g48, g2215, g2243
\c__cmd_ignore_def_tl g2468, g2486,
g2493, g2507, g2528, g2557, g2564,
g2571, g2579, g2587, g2596, g2603,
g2610, g2617, g2624, g2636, g2643, 181

```

File Key: a=ltdirname.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdblocks.dtx, j=ltalloc.dtx, k=ltcntr.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspaced.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

```

\l__cmd_last_delimiters_tl .....
..... g33, g382, g401, g527,
g541, g563, g599, g649, g659, g708, 115
\l__cmd_long_bool .....
..... g34, g386, g479, g480,
g566, g677, g688, g693, g697, g714,
g745, g862, g871, g907, g922, g935,
g952, g972, g1724, g1729, g1834, 140
\l__cmd_m_args_int .....
..... g36,
g717, g818, g846, g849, g851, g853, 115
\l__cmd_nesting_a_t1 .....
..... g1504
\l__cmd_nesting_b_t1 .....
..... g1504
\l__cmd_normalize_arg_spec:n .....
..... g91, g379, g379
\l__cmd_normalize_arg_spec_loop:n
..... g379, g391, g413, g474, g519,
g529, g544, g567, g573, g583, g589
\l__cmd_normalize_check_gv:N .....
..... g587, g635, g635
\l__cmd_normalize_check_lu:N .....
..... g635, g645
\l__cmd_normalize_E_unique-
check:w .....
..... g521, g537, g546, g551
\l__cmd_normalize_type_!:_w .....
..... g462
\l__cmd_normalize_type_+:_w .....
..... g462
\l__cmd_normalize_type_=:_w .....
..... g462
\l__cmd_normalize_type_>:_w .....
..... g462
\l__cmd_normalize_type_aux:NnNn ...
..... g462, g478, g484, g493, g508
\l__cmd_normalize_type_b:w .....
..... g591, g591
\l__cmd_normalize_type_D:w .....
..... g442, g450, g452, g521, g521
\l__cmd_normalize_type_d:w .....
..... g437, g439
\l__cmd_normalize_type_E:w .....
..... g447, g521, g531
\l__cmd_normalize_type_e:w .....
..... g437, g444
\l__cmd_normalize_type_m:w .....
..... g569, g569
\l__cmd_normalize_type_O:w .....
..... g437, g451
\l__cmd_normalize_type_o:w .....
..... g437, g449
\l__cmd_normalize_type_R:w .....
..... g456, g569, g575
\l__cmd_normalize_type_r:w .....
..... g437, g453
\l__cmd_normalize_type_s:w .....
..... g437, g458
\l__cmd_normalize_type_t:w .....
..... g459, g521, g554
\l__cmd_normalize_type_v:w .....
..... g569, g585
\l__cmd_obey_spaces_bool .....
..... g32, g385, g485, g487, g560, g565,
g694, g698, g715, g752, g863, g872, 138
\l__cmd_peek_cs_check_equal:NNN ...
..... g2432, g2446, g2452
\l__cmd_peek_meaning:NTF .....
..... g2423, g2432, g2432
\l__cmd_peek_meaning_aux:NNTF ...
..... g2432, g2433, g2435, g2436
\l__cmd_peek_meaning_remove:NTF ..
g1446, g1451, g1469, g1475, g1576,
g1582, g1709, g2425, g2432, g2434
\l__cmd_peek_nonspace:NTF g2422, g2422
\l__cmd_peek_nonspace_aux:nNNTF ..
g2422, g2423, g2425, g2426, g2429
\l__cmd_peek_nonspace_remove:NTF .
g1436, g1441, g1457, g1463, g1564,
g1570, g1697, g1707, g2422, g2424
\l__cmd_peek_true_remove>NNw .. g2432
\l__cmd_peek_true_remove:Nw .....
..... g2443, g2447, g2455, g2459
\l__cmd_prefixed_bool .....
..... g37,
g729, g746, g753, g759, g767, g816, 115
\l__cmd_prepare_signature:N .....
..... g711, g724, g727,
g779, g787, g800, g819, g827, g835,
g842, g908, g918, g960, g973, g986, 135
\l__cmd_prepare_signature:n .....
..... g92, g711, g711
\l__cmd_prepare_signature_-
bypass:N .....
..... g711,
g730, g732, g747, g754, g762, g772, 135
\l__cmd_process_all_t1 . g38, g128,
g245, g253, g318, g720, g850, g874, 116
\l__cmd_process_one_t1 . g39, g721,
g761, g770, g796, g805, g878, g881, 116
\l__cmd_process_some_bool .....
..... g40, g127, g722, g760, g769, 116
\l__cmd_put_arg_expandable:nw ...
..... g1910, g1915, g1916, g1951,
g1956, g1957, g2064, g2064, g2065
\l__cmd_replicate_processor:nn ...
..... g795, g802, g802
\l__cmd_run_code: .....
..... g247, g250, g250, g1414, g1422,
g1430, g1433, g1438, g1443, g1448,
g1454, g1460, g1466, g1472, g1560,
g1566, g1572, g1578, g1602, g1617,
g1624, g1635, g1637, g1643, g1649,
g1655, g1661, g1667, g1673, g1679,
g1690, g1692, g1710, g1732, g1871, 121
\l__cmd_saved_args_t1 .....
..... g41, g1418, g1426, 116
\l__cmd_set_environment_end:n ...
..... g211, g255
\l__cmd_set_eq_if_exist>NN g1024,
g1041, g1043, g1062, g1081, g1082,
g1083, g1096, g1097, g1098, g1184

```

File Key: a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

```

\__cmd_show:N   g1228, g1230, g1230,
                g1256, g1279, g1319, g1321, g1403, 151
\__cmd_show:n   ... g1246, g1305, g1311
\__cmd_show_arg_spec:N ...
                ..... g2316, g2316, g2888
\__cmd_show_arg_spec:n ...
                ..... g2316, g2322, g2890
\__cmd_show_command:N ...
                ..... g1236, g1246, g1246
\__cmd_show_command:NnNNwN ...
                ..... g1246, g1247, g1248
\__cmd_show_command_aux:NnNN ...
                ..... g1246,
                g1250, g1263, g1272, g1281, g1305, 150
\__cmd_show_delim:Nw ...
                .. g1351, g1351
\__cmd_show_delims:Nw ...
                .. g1351, g1353
\__cmd_show_delims_opt:Nw ...
                ..... g1351, g1355
\__cmd_show_E:Nw ...
                .. g1351, g1370
\__cmd_show_e:Nw ...
                .. g1351, g1359
\__cmd_show_environment:N ...
                ..... g1238, g1246, g1291, g1316
\__cmd_show_environment:Nnnw ...
                ..... g1293, g1301
\__cmd_show_environment_end:N ...
                ..... g1239, g1313
\__cmd_show_expandable:N ...
                ..... g1237, g1246, g1253
\__cmd_show_expandable:NnNNNNnN ...
                ..... g1246, g1254,
                g1259, g1261, g1268, g1270, g1276
\__cmd_show_opt:Nw ...
                .. g1351, g1357
\__cmd_show_prefix:Nw ...
                .. g1351, g1383
\__cmd_show_processor:Nw ...
                .. g1351, g1385
\c__cmd_show_type_!_tl ...
                .. g1345
\c__cmd_show_type_+_tl ...
                .. g1345
\c__cmd_show_type_>_tl ...
                .. g1345
\c__cmd_show_type_D_tl ...
                .. g1345
\c__cmd_show_type_d_tl ...
                .. g1345
\c__cmd_show_type_E_tl ...
                .. g1345
\c__cmd_show_type_e_tl ...
                .. g1345
\c__cmd_show_type_0_tl ...
                .. g1345
\c__cmd_show_type_R_tl ...
                .. g1345
\c__cmd_show_type_r_tl ...
                .. g1345
\c__cmd_show_type_t_tl ...
                .. g1345
\l__cmd_signature_tl ...
                ..... g42, g121, g164,
                g723, g786, g799, g826, g834, g848,
                g857, g990, g1417, g1486, g1592,
                g1602, g1619, g1626, g1635, g1639,
                g1645, g1651, g1657, g1663, g1669,
                g1675, g1681, g1712, g1734, g1871, 165
\__cmd_single_token_check:n ...
                ..... g524, g525,
                g535, g557, g578, g579, g607, g607
\l__cmd_some_long_bool ...
                .. g44, g158, g166, g389, g678, g689, 125
\l__cmd_some_obey_spaces_bool ...
                ..... g43, g388, g488, g702, 116
\l__cmd_some_short_bool ...
                .. g45, g157, g168, g390, g690, 116
\__cmd_split_add_item:n ...
                .. g1384, g1387, g1386, g1392, 153
\__cmd_split_argument:nnn ...
                ..... g2107, g2107, g2875
\__cmd_split_argument_aux:n ...
                ..... g2107, g2123, g2142
\__cmd_split_argument_aux:nnnn ...
                ..... g2107, g2110, g2114
\__cmd_split_argument_aux:wn ...
                ..... g2107, g2143, g2144
\__cmd_split_end_item:n ...
                .. g1343, g1352, g1354, g1356,
                g1358, g1364, g1375, g1387, g1394, 153
\__cmd_split_list:nn ...
                ..... g2072, g2074, g2109, g2876
\__cmd_split_list_multi:nn ...
                .. g2072, g2079, g2082, g2084, g2091, g2104
\l__cmd_split_list_seq ...
                ..... g2072, g2086, g2088, 172
\__cmd_split_list_single:Nn ...
                ..... g2072, g2080, g2094
\l__cmd_split_list_tl ...
                .. g2073, g2096, g2102, g2104, 172
\__cmd_split_N_head_apply:Nn ...
                .. g2152, g2172, g2184, g2259
\__cmd_split_N_head_apply_-aux:NNw ...
                .. g2152, g2260, g2261
\__cmd_split_signature:n ...
                .. g1283, g1322, g1322, 150
\__cmd_split_signature_loop:Nw ...
                ..... g1327, g1329,
                g1329, g1343, g1352, g1354, g1356,
                g1358, g1366, g1379, g1384, g1386
\__cmd_split_start_item: ...
                .. g1332, g1363, g1374, g1387, g1387, 151
\__cmd_start:nNNnnn ...
                .. g117, g219, g230, g2404, 180
\__cmd_start_aux:NNnnnn ...
                .. g225, g236, g239, g239, g249
\__cmd_start_env:nnnnn ...
                .. g113, g219, g219, g2406, 180
\__cmd_start_expandable:nNNNNn ...
                .. g151, g337, g337, g2405, 119

```

File Key: a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\l__cmd_suppress_strip_bool \cot I18
 g35, g387, g494, g502, g716, g768, g865, g873, 115
 \l__cmd_t1_mapthread_function:NNN \coth I19
 g288, g316, g2370, g2370
 \l__cmd_t1_mapthread_function:nnN \countdef d75, d85,
 g364, g1377, g2370, g2383, 152
 \l__cmd_t1_mapthread_loop:w \counterwithin 449
 .. g2370, g2373, g2385, g2389, g2394
 \l__cmd_tmp:w \counterwithin t75, 1144
 g49, g52, g274, \counterwithout 449
 g290, g332, g334, g437, g461, g995, aa75, b37, b38, b39, b41, b51, b90, a69
 g1001, g1019, g1213, g1223, g1345, \counterwithout t75, 1144
 g1348, g1350, g1368, g1372, g1378, \CountZero d225
 g1382, g1876, g1895, g1928, g1947, \cr s500, s506,
 g2004, g2028, g2031, g2055, g2464, 122
 s516, s522, E101, E105, E930, E934,
 \l__cmd_tmp_prop I188, I192, I378, I424, I540, L192,
 g49, g1588, g1591, g1597, 117 L203, L210, L219, L224, L230,
 \l__cmd_tmpt1 L377, M141, M143, M148, M154, b446
 g50, g286, g291, g301, g964, g966,
 g980, g983, g1124, g1130, g1145, \crcr s327, s362, s363, s390,
 g1153, g1169, g1176, g1194, g1197, s394, s397, s476, s480, s484, s486,
 g1287, g1315, g1316, g1325, g1396, s489, s732, s760, s764, s767, s834,
 g1525, g1528, g2438, g2463, g2466, 151 s837, s895, s1253, A651, B337,
 \l__cmd_tmptb_t1 B338, B340, B459, B462, B466,
 g51, g949, g954, B530, B531, B532, B533, B534,
 g965, g1195, g1197, g1326, g1332, B535, B537, B538, B539, B540,
 g1389, g1393, g1397, g1398, g1597, B541, B543, E106, E935, I168, I170,
 g1598, g1600, g2439, g2450, g2456, 117 I171, I172, I188, I190, I191, I192,
 \l__cmd_token_if_cs:NTF I210, I211, L171, L172, M141, b536
 g1549, g2361, g2361, g2445, 131
 \l__cmd_trim_spaces:n create commands:
 g2150, g2150, g2877
 \l__cmd_use_i_delimit_by_q_- \create_callback d769
 recursion_stop:nw g46, g2219
 \l__cmd_v_arg_t1 \create_callback 47
 g1721, g1738, \CS 82
 \l__cmd_v_nesting_int \cs h2680, h2681, h2683
 g1757, g1791, g1846, g1853, g1860, 162 cs commands:
 \colon B503
 \color 350
 \columnsep r27, r97, r154, Y81, Y204
 \columnseprule Y82, Y2264, Y2298
 \columnwidth r24, r27, r28, r30,
 r94, r97, r98, r100, r151, r154, r155,
 r158, K347, K386, K404, K421,
 P99, P168, P470, P489, P507,
 Y80, Y146, Y147, Y148, Y203,
 Y204, Y205, Y206, Y207, Y1847,
 Y1849, Y2262, Y2266, Y2294, Y2300
 \cong B457
 \contentsline O164, O171, O177, O184, 813
 \coprod B341
 \copyright s285, s316, A649, 1117
 \cos I12
 \cosh I14

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=ltcntr1.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

```

\cs_generate_from_arg_count:NNnn
    ..... g106, g134,
    g142, g204, g290, g293, g332, 1152
\cs_generate_variant:Nn .....
    . i20, r528, V22, V133, W462,
    W470, aa601, g249, g293, g1043,
    g1873, g2065, g2091, h37, h38,
    h39, h40, h41, h42, h43, h53, h54,
    h57, h66, h1040, h1063, h1573, h1647
\cs_gset:Npn .....
    . h2203, h2269, z285, z324, W242,
    X173, h1039, h1059, h1569, h1980
\cs_gset:Npx ..... W213, h1008
\cs_gset_eq:NN .....
    . h2210, h2211, h2212, h2213,
    h2472, i56, i544, i572, z283, z291,
    z316, z330, e136, e137, e138, e139,
    e140, e141, e142, X52, X151, X166,
    X167, X172, X181, X185, X321,
    X387, g2302, g2891, h750, h1376,
    h1402, h1415, h1416, h1419, h2110
\cs_gset_nopar:Npn ..... h2856
\cs_gset_nopar:Npx ..... h50, h52
\cs_gset_protected:Npn ... h2264,
    h2292, h2693, h2834, i231, i423,
    i550, V29, h88, h91, h100, h133,
    h160, h188, h191, h197, h221, h226,
    h251, h310, h328, h578, h580, h597,
    h621, h668, h675, h1400, h1457,
    h2056, h2058, h2086, h2091, h2109
\cs_gset_protected:Npx .....
    . S231, X20, h20, h1120
\cs_if_eq:NNTF .....
    . V64, V66, V121, g1001, g1608
\cs_if_exist:N ..... 929
\cs_if_exist:NTF ..... h2374,
    h2403, h2411, i73, i81, V123, W93,
    W439, W443, g67, g183, g1004,
    g1042, g2271, g2278, g2744, g2757,
    g2769, g2778, g2781, g2788, g2793,
    g2800, g2813, g2826, h1011, h1076, 950
\cs_if_exist_p:N ..... g74, g75, h277
\cs_if_exist_use:NTF .....
    . e181, X313, X314, X318, X319,
    g417, g1147, h1242, h1286, h1311
\cs_if_free:NTF ..... V73
\cs_new:Npn .. h2181, h2190, h2225,
    h2401, h2409, h2423, h2441, h2450,
    h2455, h2701, h2702, h2703, h2704,
    h2708, h2709, i303, i313, i323, i325,
    i327, i338, i361, i362, i365, i485,
    i598, n78, n81, n106, n118, r519,
    e179, S160, S166, S167, S168, V130,
    V132, W30, W38, W44, W57, W95,
    W100, W105, W112, W127, W234,
    W240, W412, W414, W416, W419,
    W423, W437, W453, W463, W583,
    X54, X59, X77, X141, X146, X163,
    X178, X183, X189, X192, X206,
    X257, X310, X323, X337, X340,
    X350, X392, X511, X512, X513,
    g337, g342, g344, g346, g358, g370,
    g376, g1066, g1109, g1190, g1215,
    g1220, g1539, g1874, g1878, g1896,
    g1903, g1904, g1926, g1930, g1948,
    g1961, g1963, g1965, g1967, g1973,
    g1990, g1992, g1994, g1996, g1998,
    g2000, g2002, g2006, g2029, g2033,
    g2056, g2058, g2064, g2142, g2144,
    g2169, g2175, g2181, g2187, g2259,
    g2261, g2370, g2383, g2389, g2470,
    g2835, g2852, g2853, g2857, g2858,
    g2859, h45, h46, h330, h336, h349,
    h359, h360, h362, h376, h382, h874,
    h876, h878, h1161, h1173, h1178,
    h1186, h1195, h1197, h1204, h1231,
    h1238, h1240, h1348, h1364, h1487,
    h1488, h1649, h1650, h1976, h1983,
    h2119, h2144, h2152, h2165, h2173
\cs_new_eq:NN ..... h2180,
    h2722, h2723, h2793, h2795, i13,
    i14, i440, n128, e159, e160, e161,
    e162, e178, S218, S234, S235, S271,
    S273, S368, S370, V8, W252, W253,
    W510, W512, W514, W516, W518,
    W520, W522, W524, W526, W528,
    X7, X152, X347, X349, X411,
    X414, X415, X510, aa629, aa630,
    g52, g97, g1165, g1166, g1173,
    g1180, g2301, g2738, g2739, g2854,
    g2855, g2856, g2862, g2863, g2864,
    g2874, g2875, g2876, g2877, g2878,
    g2884, g2890, h7, h23, h34, h63,
    h1145, h1327, h1333, h1371, h1724,
    h1725, h1726, h1727, h1728, h1729
\cs_new_protected:Npn ..... h2178, h2185, h2208, h2231,
    h2240, h2251, h2256, h2274, h2303,
    h2305, h2315, h2457, h2463, h2469,
    h2635, h2636, h2637, h2660, h2671,
    h2711, h2712, h2713, h2714, h2724,
    h2731, h2738, h2745, h2752, h2754,
    h2756, h2763, h2770, h2777, h2784,
    i18, i23, i25, i34, i36, i46, i54,
    i63, i77, i97, i117, i130, i135, i144,
    i149, i156, i296, i367, i404, i441,
    i469, i498, n86, z278, S7, S17, S46,
    S112, S123, S131, S134, S219, S224,
```

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=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphe.dtx, aa=ltfinal.dtx

S229, S237, S265, S283, S314, V25, V27, V62, V71, V90, V102, V111, V119, V134, V138, V156, V170, W49, W62, W70, W79, W208, W222, W393, W398, W421, W548, X8, X13, X18, X343, X473, aa582, aa610, aa615, aa620, aa628, g55, g60, g65, g86, g98, g104, g132, g178, g197, g211, g219, g239, g250, g260, g271, g279, g284, g294, g299, g313, g322, g330, g379, g413, g439, g444, g449, g451, g453, g458, g462, g476, g482, g491, g508, g521, g531, g546, g554, g569, g575, g585, g591, g607, g616, g635, g645, g657, g668, g673, g674, g700, g711, g727, g732, g742, g749, g756, g764, g774, g781, g789, g802, g813, g821, g829, g837, g844, g855, g883, g893, g898, g905, g910, g912, g920, g933, g946, g962, g968, g975, g977, g988, g993, g999, g1017, g1024, g1041, g1044, g1059, g1078, g1093, g1126, g1134, g1142, g1151, g1163, g1167, g1174, g1179, g1181, g1192, g1199, g1207, g1230, g1246, g1248, g1253, g1261, g1270, g1281, g1291, g1301, g1311, g1313, g1322, g1329, g1351, g1353, g1355, g1357, g1359, g1370, g1383, g1385, g1387, g1392, g1394, g1406, g1408, g1410, g1412, g1414, g1422, g1433, g1438, g1443, g1448, g1453, g1459, g1465, g1471, g1477, g1484, g1507, g1560, g1566, g1572, g1578, g1584, g1606, g1616, g1617, g1624, g1637, g1643, g1649, g1655, g1661, g1667, g1673, g1679, g1690, g1692, g1694, g1706, g1708, g1710, g1722, g1727, g1732, g1753, g1759, g1769, g1775, g1788, g1801, g1807, g1830, g1838, g1858, g1866, g1868, g2066, g2074, g2084, g2094, g2107, g2114, g2150, g2152, g2157, g2193, g2198, g2209, g2211, g2213, g2230, g2241, g2251, g2253, g2255, g2257, g2267, g2273, g2280, g2292, g2304, g2309, g2316, g2322, g2329, g2333, g2361, g2396, g2416, g2422, g2424, g2426, g2432, g2434, g2436, g2452, g2459, g2740, g2753, g2766, g2771, g2776, g2786, g2792, g2794, g2796, g2809, g2822, g2830, g2879, g2885, h8, h13, h18, h47, h49, h51, h55, h58, h64, h69, h71, h73, h104, h148, h172, h174, h176, h201, h203, h205, h231, h266, h268, h285, h290, h292, h383, h392, h397, h405, h429, h431, h447, h455, h465, h474, h485, h491, h497, h513, h526, h552, h569, h625, h644, h651, h658, h685, h690, h695, h704, h742, h744, h752, h754, h757, h759, h927, h929, h962, h995, h1007, h1009, h1038, h1051, h1053, h1055, h1057, h1074, h1098, h1105, h1262, h1269, h1300, h1322, h1328, h1334, h1339, h1342, h1345, h1346, h1374, h1425, h1496, h1577, h1651, h1658, h1667, h1674, h1681, h1688, h1696, h1702, h1713, h1730, h1737, h1749, h1754, h1759, h1761, h1765, h1881, h1994, h2009, h2014, h2023, h2041, h2047, h2060, h2073, h2100, h2104, h2114, h2127, h2136, h2150, h2156, 979
\cs_new_protected:Npx
..... i475, g230, g1685, g1794
\cs_new_protected_nopar:Npn
..... aa625, g1541, g1631
\cs_parameter_spec:N h1189
\cs_prefix_spec:N i207, i282, e138
\cs_replacement_spec 249
\cs_replacement_spec:N e140,
..... g1288, g1297, h37, h41, h1164,
..... h1392, h1412, h1805, h1818, h1827
\cs_set:Npn h2461, h2467,
..... h2843, i199, i274, i456, i520, i556,
..... n61, g144, g205, g290, g332, g995,
..... h1107, h1118, h1124, h1133, h1802, 153
\cs_set:Npx
..... W211, W225, g173, g923, g936
\cs_set_eq:NN i171,
..... i172, i192, i193, i198, i246, i247,
..... i267, i268, i273, i452, i453, i516,
..... i517, i552, i553, i611, n84, n85,
..... n129, n130, n131, n133, n134, n135,
..... n169, n170, n171, n174, z294, z333,
..... S161, S162, S163, V52, X24, X44,
..... X83, X95, X125, X132, X417, X419,
..... X421, X423, X425, g207, g208,
..... g1019, g1042, g1061, g1080, g1095,
..... g1158, g1183, g1188, g1203, g1205,
..... g1640, g1646, g1652, g1658, g1664,
..... g1670, g1676, g1682, g1832, g2464,
..... h1442, h1443, h1444, h1445, h1474,
..... h1475, h1476, h1477, h1751, h1756
\cs_set_nopar:Npn aa584, 153
\cs_set_nopar:Npx
..... g149, g170, g175, g202, g213,
..... g924, g937, g1086, g1100, g1201, h48

File Key: a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

\cs_set_protected:Npn	i173	\cyra	s1508, s1559
i248, z311, z343, V5, V31, V45,		\CYRABHCH	s1508
V144, X46, X135, X412, aa599,		\cyrabchch	s1508
g108, g437, g1213, g1345, g1368,		\CYRABHCHDSC	s1508
g1372, g1409, g1413, g1440, g1450,		\cyrabchchdsc	s1508
g1462, g1474, g1569, g1581, g1593,		\CYRABHDZE	s1509
g1627, g1693, g1713, g1876, g1928,		\cyrabhdze	s1508
g2004, g2031, h418, h1996, h2026		\CYRABHHA	s1509
\cs_set_protected_nopar:Npn		\cyrabhha	s1509
g1407, g1411, g1435, g1445, g1456,		\CYRAE	s1509
g1468, g1563, g1575, g1620, g1691		\cyrae	s1509
\cs_set_protected_nopar:Npx		\CYRB	s1509
g109, g149, g1063, g1086, g1186		\cyrb	s1509
\cs_to_str:N		\CYRBYUS	s1510
i112, i133, i147, i152, e136,		\cyrbyus	s1509
g74, g75, g88, g1035, g1037, g1242,		\CYRC	s1510
g1296, g1298, g1315, g1552, g2411, 232		\cycrc	s1510
\cs_undefine:N		\CYRCH	s1510
.. h2218, h2299, h2437, i59, i210,		\cyrch	s1510
i494, i507, W227, g2868, g2869,		\CYRCHLDSC	s1510
g2870, g2892, h79, h80, h90, h132,		\cyrchldsc	s1510
h190, h196, h271, h642, h1044,		\CYRCHRDSC	s1511
h1045, h1046, h1047, h1067, h1068,		\cyrchrdsc	s1510
h1069, h1070, h1094, h1149, h1169,		\CYRCHVCRS	s1511
h1227, h1248, h1249, h1260, h1347		\cyrchvcrs	s1511
cs\check@icr commands:		\CYRD	s1511
\cs_gset_eq:NN	77	\cyrd	s1511
\csc	I21	\CYRDELTA	s1511
\csname	681	\cyrdelta	s1511
\csname\endcsname	948	\CYRDJE	s1512
\cup	B373	\cyrdje	s1512
\CurrentFile	U853,	\CYRDZE	s1512
W3, W67, W84, W166, W171,		\cyrdze	s1512
W174, W378, W382, W556, W557, 934		\CYRDZHE	s1512
\CurrentFilePath	W3,	\cyrdzhe	s1512
W67, W83, W165, W379, W382, 934		\CYRE	s1512
\CurrentFilePathUsed	W3,	\cyre	s1512
W66, W81, W163, W376, W379, 934		\CYREPS	s1513
\CurrentFileUsed U853, W3, W66, W82,		\cyreps	s1512
W164, W376, W378, W555, W556, 934		\CYREREV	s1513
\CurrentOption		\cyrerev	s1513
s1536, s1539, s1544, s1556,		\CYRERY	s1513
U14, U457, U468, U481, U482,		\cyrery	s1513
U487, U500, U501, U503, U517,		\CYRF	s1513
U518, U521, U535, U540, U541,		\cyrf	s1513
U554, U559, U560, U573, U575,		\CYRFITA	s1514
U581, U583, U595, U596, U597,		\cyrfita	s1513
U605, U606, U607, U903, U968,		\CYRG	s1514
U1066, U1067, U1077, V49, V50, 1148		\cyrg	s1514
CurrentOption commands:		\CYRGDSC	s1514
\CurrentOption:		\cyrgdsc	s1514
.. U480, U499, U516, U534,		\CYRGDSCHCRS	s1514
U539, U553, U558, U594, U604, U1076		\cyrgdschcrs	s1514
\CYRA	s1508	\CYRGHCRS	s1515

File Key: a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphe.dtx, aa=ltfinal.dtx

\cyrgcrs	s1515	\cyrmhk	s1521
\CYRGHK	s1515	\CYRN	s1522
\cyrghk	s1515	\cyrn	s1522
\CYRGUP	s1515	\CYRNDSC	s1522
\cyrgup	s1515	\cyrndsc	s1522
\CYRH	s1515	\CYRNG	s1522
\cyrh	s1515	\cyrng	s1522
\CYRHDSC	s1516	\CYRNHK	s1522
\cyrhdsc	s1516	\cyrnhk	s1522
\CYRHCRS	s1516	\CYRNJE	s1523
\cyrhhcrs	s1516	\cyrnje	s1522
\CYRHHK	s1516	\CYRNLHK	s1523
\cyrhhk	s1516	\cyrnlhk	s1523
\CYRHDSN	s1517	\CYRO	s1523
\cyrhdsn	s1516	\cyro	s1523
\CYRI	s1517	\CYROTLD	s1523
\cyri	s1517	\cyrotld	s1523
\CYRIE	s1517	\CYRP	s1523
\cyrie	s1517	\cyrp	s1523
\CYRII	s1517	\CYRPHK	s1524
\cyrii	s1517	\cyrphk	s1524
\CYRISHRT	s1517	\CYRQ	s1524
\cyrishrt	s1517	\cyrq	s1524
\CYRISHRTDSC	s1518	\CYRR	s1524
\cyrishrtdsc	s1518	\cyrr	s1524
\CYRIZH	s1518	\CYRRDSC	s1524
\cyrizh	s1518	\cyrrdsc	s1524
\CYRJE	s1518	\CYRRHK	s1525, 1124
\cyrje	s1518	\cyrrhk	s1524, 1124
\CYRK	s1518	\CYRRHOOK	1124
\cyrk	s1518	\cyrrhook	1124
\CYRKBEAK	s1519	\CYRRTICK	s1525
\cyrkbeak	s1519	\cyrrtick	s1525
\CYRKDSC	s1519	\CYRS	s1525
\cyrkdsc	s1519	\cyrs	s1525
\CYRKHCRS	s1519	\CYRSACRS	s1525
\cyrkhcrs	s1519	\crysacrs	s1525
\CYRKHK	s1520	\CYRSCHWA	s1526
\cyrkhk	s1519	\cryschwa	s1526
\CYRKVCRS	s1520	\CYRSDSC	s1526
\cyrkvcrs	s1520	\crysdesc	s1526
\CYRL	s1520	\CYRSEMISFTSN	s1526
\cyrl	s1520	\cyrsemisftsn	s1526
\CYRLDSC	s1520	\CYRSFTSN	s1527
\cyrldesc	s1520	\crysftsn	s1527
\CYRLHK	s1521	\CYRSH	s1527
\cyrlhk	s1520	\cyrsh	s1527
\CYRLJE	s1521	\CYRSHCH	s1527
\cyrlje	s1521	\cyrshch	s1527
\CYRM	s1521	\CYRSHHA	s1527
\cyrm	s1521	\cyrshha	s1527
\CYRMDSC	s1521	\CYRT	s1528
\cyrmdsc	s1521	\cyrt	s1528
\CYRMHK	s1521	\CYRTDSC	s1528

File Key: a=ltdirname.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\cyrtdesc	s1528	\dblfloatpagefraction	P287, P301, Y2340
\CYRTETSE	s1528	\dblfloatsep	... Y757, Y769, Y1644, Y1770, Y2347
\cyrtetse	s1528	\dbltextfloatsep	... Y224, Y232, Y773, Y1643, Y1769, Y2347
\CYRTSHE	s1528	\dbltopfraction	... P284, P298, Y2339
\cyrtshe	s1528	\ddag	s313, 1104
\CYRU	s1529	\ddagger	s313, t166, t172, t179, t181, B374
\cyrus	s1529	\ddot	... B518
\CYRUSHRT	s1529	\ddots	... B513
\cyrushrt	s1529	\deadcycles	... r335, r391, r430, H32, H106, H172, Y301, 1120
\CYRV	s1529	debug commands:	
\cyrv	s1529	\debug_resume:	... 220
\CYRW	s1529	\debug_suspend:	... 220
\cyrw	s1529	\DebugHooksOff	... h2713 , h2819, 203
\CYRY	s1529	\DebugHooksOn	... h2713 , h2818, 203
\cypy	s1529	\DebugMarksOff	... S234 , 851
\CYRYA	s1530	\DebugMarksOn	... S234 , 851
\cypyra	s1530	\DebugShipoutsOff	... X414 , X446, 961
\CYRYAT	s1530	\DebugShipoutsOn	... X414 , X445, 961
\cypyat	s1530	declare commands:	
\CYRYHCRS	s1530	declare_callback_rule	... d860
\cypyhcrcs	s1530	\declare_callback_rule	... 47
\CYRYI	s1530	\DeclareAccent	... 1088
\cypyi	s1530	\DeclareCaseChangeEquivalent	aa576, 1148
\CYRYO	s1531	\DeclareCommandCopy	f472, f473, f475, 1138
\cypyo	s1530	\DeclareCurrentRelease	E812, E815, U1649
\CYRYU	s1531	\DeclareDefaultHookRule	h2717 , h2822, 201
\cypyu	s1531	\DeclareDocumentCommand	... g2740
\CYRZ	s1531	\DeclareDocumentEnvironment	... g2776
\cypyz	s1531	\DeclareEmphSequence	... 588
\CYRDSC	s1531	\DeclareEmphSequence	... A541 , A577, A578, A590, 589
\cypydsc	s1531	\DeclareEncoding	... 1088
\CYRZH	s1531	\DeclareEncodingSubset	... E40 , E397, E398,
\cypyzh	s1531	... E399, E400, E401, E402, E403,	
\CYRHDSC	s1532	... E404, E405, E406, E407, E408,	
\cypyhdsc	s1532	... E409, E410, E411, E412, E413,	
D		... E414, E415, E416, E417, E418,	
\d	s235, s395, s482, s765, s1249, s1463, s1464, s1465, s1466, s1469, s1470, s1471, s1472, s1473, s1474, s1475, s1476, s1477, s1478, s1479, s1480, s1481, s1482, s1483, s1484, s1485, s1486, s1487, s1488, s1489, s1490, s1491, s1492, s1493, s1494, s1495, s1496, s1497, s1498, s1499, s1500, s1501, s1502, 1108	... E419, E420, E421, E422, E423,	
\dag	s312, 1104	... E424, E425, E427, E428, E429,	
\dagger	s312, t165, t171, t179, t180, B375	... E430, E431, E432, E433, E434,	
\dashbox	M309, M808, M825	... E435, E436, E437, E438, E439,	
\dashv	B403	... E440, E441, E442, E443, E444,	
\date	803	... E445, E446, E447, E448, E449,	
\date	O9, O25	... E450, E451, E452, E453, E454,	
\day	a188, U1192, U1325, U1414, b384	... E455, E456, E457, E458, E459,	
\dblfigrule	Y771, Y2363 , 1095	... E460, E461, E462, E463, E464,	
		... E465, E466, E467, E468, E469,	
		... E470, E471, E472, E473, E474,	
		... E475, E476, E477, E478, E479,	
		... E480, E481, E482, E483, E484,	

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

E485, E486, E487, E488, E489,
 E490, E491, E492, E493, E494,
 E495, E496, E497, E498, E499,
 E500, E501, E502, E503, E504,
 E505, E506, E507, E508, E509,
 E510, E511, E512, E513, E514,
 E515, E516, E517, E518, E519,
 E520, E521, E522, E523, E524,
 E525, E526, E527, E528, E529,
 E530, E531, E532, E533, E534,
 E535, E536, E537, E538, E539,
 E540, E541, E542, E543, E544,
 E545, E546, E547, E548, E549,
 E550, E551, E552, E553, E554,
 E555, E556, E557, E558, E559,
 E560, E561, E562, E563, E564,
 E565, E566, E567, E568, E569,
 E570, E571, E572, E573, E574,
 E575, E620, E820, E821, E822,
 E823, E865, E872, E873, E874,
 E875, E1151, E1152, E1153, E1154,
 E1155, E1156, E1157, E1158,
 E1159, E1160, E1161, E1162,
 E1163, E1164, E1165, E1166,
 E1167, E1168, E1169, E1170,
 E1171, E1172, E1173, E1174,
 E1175, E1176, E1177, E1178,
 E1179, E1180, E1181, E1182,
 E1183, E1184, E1185, E1186,
 E1187, E1188, E1189, E1190,
 E1191, E1192, E1193, E1194,
 E1195, E1196, E1197, E1198,
 E1199, E1200, E1201, E1202,
 E1203, E1204, E1205, E1206,
 E1207, E1208, E1209, E1210, E1211
`\DeclareEnvironmentCopy`
 f513, f514, f516, 1150
`\DeclareErrorFont` v502, z383, A735, B49
`\DeclareExpandableDocumentCommand` g2796
`\DeclareFixedFont` v75
`\DeclareFontEncoding` s377,
 s463, s716, s738, s744, s830, s1038,
 v118, B126, B127, B128, B129, 1088
`\DeclareFontEncodingDefaults`
 v168, y90, y91, B36
`\DeclareFontFamily` ... v93, y85, y86, 475
`\DeclareFontFamilySubstitution` .. v462
`\DeclareFontSeriesChangeRule`
 w3, w4, w6, w7, w8,
 w9, w10, w11, w12, w13, w14, w15,
 w16, w17, w18, w19, w20, w21,
 w22, w23, w24, w25, w26, w27,
 w28, w29, w30, w31, w32, w33,
 w34, w35, w36, w37, w38, w39,
 w40, w41, w42, w43, w44, w45,
 w46, w47, w48, w49, w50, w51,
 w52, w53, w54, w55, w56, w57,
 w58, w59, w60, w61, w62, w63,
 w64, w65, w66, w67, w68, w69,
 w70, w71, w72, w73, w74, w75,
 w76, w77, w78, w79, w80, w81,
 w82, w83, w84, w85, w86, w87,
 w88, w89, w90, w91, w92, w93,
 w94, w95, w96, w97, w98, w99,
 w100, w101, w102, w103, w104,
 w105, w106, w107, w108, w109,
 w110, w111, w112, w113, w114,
 w115, w116, w117, w118, w119,
 w120, w121, w122, w123, w124,
 w125, w126, w127, w128, w129,
 w130, w131, w132, w133, w134,
 w135, w136, w137, w138, w139,
 w140, w141, w142, w143, w144,
 w145, w146, w147, w148, w149,
 w150, w151, w152, w153, w154,
 w155, w156, w157, w158, w159,
 w160, w161, w162, w163, w164,
 w165, w166, w167, w168, w169,
 w170, w171, w172, w173, w174,
 w175, w176, w177, w178, w179,
 w180, w181, w182, w183, w184,
 w185, w186, w187, w188, w189,
 w190, w191, w192, w193, w194,
 w195, w196, w197, w198, w199,
 w200, w201, w202, w203, w204,
 w205, w206, w207, w208, w209,
 w210, w211, w212, w213, w214,
 w215, w216, w217, w218, w219,
 w220, w221, w222, w223, w224,
 w225, w226, w227, w228, w229,
 w230, w231, w232, w233, w234,
 w235, w236, w237, w238, w239,
 w240, w241, w242, w243, w244,
 w245, w246, w247, w248, w249,
 w250, w251, w252, w253, w254,
 w255, w256, w257, w258, w259,
 w260, w261, w262, w263, w264,
 w265, w266, w267, w268, w269,
 w270, w271, w272, w273, w274,
 w275, w276, w277, w278, w279,
 w280, w281, w282, w283, w284,
 w285, w286, w287, w288, w289,
 w290, w291, w292, w293, w294,
 w295, w296, w297, w298, w299,
 w300, w301, w302, w303, w304,
 w305, w306, w307, w308, w309,
 w310, w311, w312, w313, w314,
 w315, w316, w317, w318, w319,

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

w320, w321, w322, w323, w324,
 w325, w326, w327, w328, w329,
 w330, w331, w332, w333, w334,
 w335, w336, w337, w338, w339,
 w340, w341, w342, w343, w344,
 w345, w346, w347, w348, w349,
 w350, w351, w352, w353, w354,
 w355, w356, w357, w358, w359,
 w360, w361, w362, w363, w364,
 w365, w366, w367, w368, w369,
 w370, w371, w372, w373, w374,
 w375, w376, w377, w378, w379,
 w380, w381, w382, w386, w388, 487
 \DeclareFontSeriesDefault 571
 \DeclareFontSeriesDefault
 A34, A35, A69,
 A71, A72, A82, A93, A102, A104, 574
 \DeclareFontShape v18, v471,
 v472, v473, v474, v475, v476, v477,
 v478, v479, v480, v481, v482, v483,
 v484, v485, v486, v487, v488, v489,
 v490, v491, y25, y27, y81, y82, 1082
 \DeclareFontShapeChangeRule
 .. w523, w524, w541, w542, w543,
 w544, w545, w546, w547, w548,
 w549, w550, w551, w552, w553,
 w554, w555, w556, w557, w558,
 w559, w560, w561, w562, w563,
 w564, w565, w566, w567, w568,
 w569, w570, w571, w572, w573,
 w574, w575, w576, w577, w578,
 w579, w580, w581, w582, w583,
 w584, w585, w586, w587, w588,
 w589, w590, w591, w592, w593,
 w594, w595, w596, w597, w601, w603
 \DeclareFontSubstitution
 s745, s831, v141, B26, B34,
 B37, B38, B130, B131, B132, B133
 \DeclareHookRule
 h2715, h2821, H46, H47, 196
 \DeclareHookrule 200
 \DeclareKeys V140, 925
 \DeclareLowercaseMapping ... aa576, 1152
 \DeclareMathAccent . z788, B516, B517,
 B518, B519, B520, B521, B522,
 B523, B524, B525, B526, B527, B528
 \DeclareMathAlphabet
 y119, y123, y125, y132,
 z614, z777, B151, B152, B153, B154
 \DeclareMathAlphabetCharacter ... z963
 \DeclareMathDelimiter
 .. z965, B255, B256, B257, B258,
 B259, B260, B263, B265, B266,
 B563, B565, B567, B569, B571,
 B574, B576, B578, B580, B582,
 B584, B586, B588, B590, B592,
 B594, B596, B598, B600, B602,
 B604, B606, B608, B610, B612, 615
 \DeclareMathOperator 1104
 \DeclareMathRadical z1100, B529
 \DeclareMathSizes
 ... v205, v211, v233, B157, B158,
 B159, B160, B161, B162, B163,
 B164, B165, B166, B167, B168, 1106
 \DeclareMathSizes* v205
 \DeclareMathSymbol .. z901, z964, z981,
 B169, B170, B171, B172, B173,
 B174, B175, B176, B177, B178,
 B179, B180, B181, B182, B183,
 B184, B185, B186, B187, B188,
 B189, B190, B191, B192, B193,
 B194, B195, B196, B197, B198,
 B199, B200, B201, B202, B203,
 B204, B205, B206, B207, B208,
 B209, B210, B211, B212, B213,
 B214, B215, B216, B217, B218,
 B219, B220, B221, B222, B223,
 B224, B225, B226, B227, B228,
 B229, B230, B231, B232, B233,
 B234, B235, B236, B237, B238,
 B239, B240, B241, B242, B243,
 B244, B245, B246, B247, B248,
 B249, B250, B251, B261, B262,
 B264, B268, B269, B270, B271,
 B272, B273, B274, B275, B276,
 B277, B278, B279, B280, B281,
 B282, B283, B284, B285, B286,
 B287, B288, B289, B290, B291,
 B292, B293, B294, B295, B296,
 B297, B298, B299, B300, B301,
 B302, B303, B304, B305, B306,
 B307, B308, B309, B310, B311,
 B312, B313, B314, B315, B316,
 B317, B318, B319, B320, B321,
 B322, B323, B324, B325, B327,
 B328, B329, B330, B331, B332,
 B333, B334, B341, B342, B343,
 B344, B345, B346, B347, B349,
 B350, B351, B352, B353, B354,
 B356, B357, B358, B359, B360,
 B361, B364, B365, B366, B367,
 B370, B371, B372, B373, B374,
 B375, B376, B377, B378, B379,
 B380, B381, B382, B383, B384,
 B385, B386, B387, B388, B389,
 B390, B391, B392, B393, B394,
 B395, B396, B397, B398, B399,
 B400, B401, B402, B403, B404,

File Key: a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

B405, B406, B407, B408, B409,
 B410, B411, B414, B415, B418,
 B419, B420, B421, B422, B423,
 B424, B425, B426, B427, B428,
 B429, B430, B432, B433, B434,
 B435, B436, B437, B438, B441,
 B442, B443, B445, B446, B447,
 B448, B449, B450, B451, B452,
 B453, B454, B455, B477, B479,
 B501, B502, B503, B553, B554,
 B555, B556, B614, B615, B616, 1120
 \DeclareMathVersion z396, A2, A3
 \DeclareOldFontCommand D125, D141
 \DeclareOption 877
 \DeclareOption .. s1535, x29, x37, x45,
 x53, x56, x60, E820, E821, E822,
 E823, E824, E826, E828, E830,
 E831, E872, E873, E874, E875,
 E876, E878, E879, U435, U1087, 1110
 \DeclareOption* 877
 \DeclareOption* U435
 \DeclarePreloadSizes
 v185, y95, y96, C19, C21,
 C22, C23, C25, C26, C27, C28,
 C29, C30, C34, C38, C43, C45,
 C49, C50, C53, C54, C57, C58, C64
 \DeclareProtectedCommand 1098
 \DeclareRelease E811, E814, U1556
 DeclareRelease commands:
 \DeclareRelease: U1559
 \DeclareRobustCommand 14, 111,
 l30, l57, p7, p8, p9, p10, p11, p69,
 p93, p331, p407, p421, p438, p463,
 q2, q3, q13, r469, r616, s150, s158,
 s307, s310, s311, s312, s313, s314,
 s316, s318, s320, t189, u19, u20,
 u21, v251, v279, v280, v281, v286,
 v298, v308, v316, v318, v336, v666,
 v675, w394, w398, w407, w409,
 w419, w526, w531, w536, w615,
 w618, w626, w627, w633, w713,
 x115, x164, A4, A7, A10, A13, A16,
 A19, A22, A25, A28, A254, A277,
 A307, A328, A352, A363, A380,
 A383, A405, A410, A415, A448,
 A453, A458, A474, A477, A480,
 A483, A486, A500, A549, A550,
 A585, A591, A613, A615, A624,
 A626, A633, A649, A657, A675,
 A691, A708, B335, B336, B337,
 B348, B355, B412, B413, B444,
 B456, B460, B463, B468, B470,
 B472, B475, B478, B480, B481,
 B483, B485, B487, B489, B491,
 B493, B495, B497, B499, B505,
 B507, B509, B512, B530, B533,
 B536, B540, B544, B547, B550,
 B557, B560, B617, B618, B619,
 B624, B626, B628, B630, D3, D126,
 E4, E11, E610, E1136, G147, G158,
 H179, H235, H364, H369, H374,
 H384, H388, H392, H473, H477, I3,
 I4, I5, I6, I7, I8, I9, I10, I11, I12,
 I13, I14, I15, I16, I17, I18, I19, I20,
 I21, I22, I23, I24, I25, I26, I27, I28,
 I29, I30, I31, I32, I33, I34, I35, I39,
 I41, I42, I43, I44, I45, I46, I47, I48,
 I49, I50, I51, I52, I81, I82, I83,
 I84, I126, I167, I169, I173, I218,
 I220, I222, I224, I227, I228, I230,
 I237, I239, I240, I251, I274, I276,
 I292, I303, I353, I354, I355, I429,
 I463, I487, K7, K24, K120, K133,
 K156, K157, K173, K184, K238,
 K437, K455, K463, K499, K500,
 K501, K502, K503, K504, L139,
 L142, L154, M124, M127, M130,
 O7, O8, O9, O10, O14, O222, P402,
 P423, R16, R28, S375, S376, T23,
 T34, T51, T59, T82, T84, T86,
 T93, U1043, U1044, U1045, U1051,
 U1052, U1679, f233, W148, W184,
 X448, Y87, f833, f834, f840, f855, 1099
 \DeclareSizeFunction x417, x490,
 x491, x502, x503, x507, x508, x514,
 x515, x543, x557, x558, x565, x566
 \DeclareSymbolFont
 . y136, z455, B141, B142, B143, B144
 \DeclareSymbolFontAlphabet
 z1174, B148, B149, B150
 \DeclareTextAccent s64,
 s378, s379, s380, s381, s382, s383,
 s384, s385, s386, s387, s388, s464,
 s465, s466, s467, s468, s469, s470,
 s471, s472, s473, s474, s741, s746,
 s747, s748, s749, s750, s751, s752,
 s753, s754, s755, s756, s838, s839,
 s840, s841, s842, s843, s844, s845,
 s846, s847, s848, s849, s850, s851, s852
 \DeclareTextAccentDefault
 s185, s226, s227, s228, s229,
 s230, s231, s232, s233, s234, s235,
 s236, s237, s238, s239, s279, s282,
 E687, E688, E940, E941, E942,
 E943, E944, E945, E946, E947,
 E948, E949, E950, E951, E952, 1103
 \DeclareTextCommand s3,
 s58, s65, s83, s389, s392, s395, s409,

File Key: a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

s410, s411, s414, s415, s422, s424,
 s426, s428, s434, s436, s438, s445,
 s475, s478, s482, s485, s487, s490,
 s492, s494, s510, s568, s569, s570,
 s730, s757, s759, s762, s765, s798,
 s805, s832, s835, s865, s893, s1053,
 s1080, E376, E377, E378, E379,
 E380, E381, E382, E383, E384,
 E385, E625, E632, E639, E759, 1097
\DeclareTextCommandDefault
 s57, s186, s188,
 s283, s286, s287, s288, s290, s292,
 s296, s300, s301, s303, s304, s305,
 s306, s326, s355, E155, E157, E160,
 E162, E164, E166, E168, E170,
 E172, E174, E176, E178, E180,
 E182, E184, E186, E188, E190,
 E193, E194, E195, E196, E197,
 E198, E199, E200, E201, E202,
 E203, E204, E205, E206, E207,
 E208, E210, E212, E214, E216,
 E218, E220, E222, E224, E226,
 E228, E230, E232, E234, E236,
 E238, E240, E242, E244, E246,
 E248, E250, E252, E254, E256,
 E258, E260, E262, E264, E266,
 E268, E270, E272, E274, E276,
 E278, E280, E282, E284, E286,
 E288, E290, E292, E294, E296,
 E298, E300, E302, E304, E306,
 E309, E311, E313, E315, E317,
 E319, E321, E323, E325, E327,
 E329, E331, E333, E335, E337,
 E339, E341, E343, E345, E347,
 E349, E351, E353, E355, E357,
 E359, E361, E363, E365, E654,
 E662, E664, E670, E678, E1006,
 E1008, E1009, E1011, E1013,
 E1015, E1017, E1019, E1021,
 E1023, E1025, E1027, E1029,
 E1031, E1033, E1035, E1037,
 E1039, E1041, E1043, E1045,
 E1047, E1049, E1051, E1053,
 E1055, E1057, E1059, E1061,
 E1063, E1065, E1067, E1069,
 E1071, E1073, E1075, E1077,
 E1079, E1081, E1083, E1085,
 E1087, E1089, E1091, E1093,
 E1095, E1097, E1099, E1101,
 E1103, E1105, E1107, E1109,
 E1111, E1113, E1115, E1117,
 E1119, E1121, E1123, E1126, 1103
\DeclareTextComposite
 s76, s452, s453, s587,
 s588, s589, s590, s591, s592, s593,
 s594, s595, s596, s597, s598, s599,
 s600, s601, s602, s603, s604, s605,
 s606, s607, s608, s609, s610, s611,
 s612, s613, s614, s615, s616, s617,
 s618, s619, s620, s621, s622, s623,
 s624, s625, s626, s627, s628, s629,
 s630, s631, s632, s633, s634, s635,
 s636, s637, s638, s639, s640, s641,
 s642, s643, s644, s645, s646, s647,
 s648, s649, s650, s651, s652, s653,
 s654, s655, s656, s657, s658, s659,
 s660, s661, s662, s663, s664, s665,
 s666, s667, s668, s669, s670, s671,
 s672, s673, s674, s675, s676, s677,
 s678, s679, s680, s681, s682, s683,
 s684, s685, s686, s687, s688, s689,
 s690, s691, s692, s693, s694, s695,
 s696, s697, s812, s813, s814, s815,
 s816, s817, s818, s819, s820, s821,
 s822, s823, s824, s825, s826, s827, 1112
\DeclareTextCompositeCommand
 s76, s431, s454, s455,
 s456, s457, s459, s698, s699, s701,
 s704, s705, s706, s707, s708, s709,
 s710, s711, s712, s795, s1059, 1104
\DeclareTextFontCommand
 D1, D15, D16, D17,
 D18, D19, D20, D21, D22, D23,
 D24, D29, D30, D31, D42, D140, 1121
\DeclareTextFoo 1088
\DeclareTextGlyph 1095
\DeclareTextSymbol s3,
 s398, s399, s400, s401, s402, s403,
 s404, s405, s406, s407, s408, s412,
 s413, s416, s417, s418, s419, s420,
 s421, s526, s527, s528, s529, s530,
 s531, s532, s533, s534, s535, s536,
 s537, s538, s539, s540, s541, s543,
 s544, s545, s546, s547, s548, s549,
 s550, s551, s552, s553, s554, s555,
 s556, s557, s558, s559, s560, s561,
 s562, s563, s564, s565, s566, s567,
 s571, s572, s573, s574, s575, s576,
 s577, s578, s579, s580, s581, s582,
 s583, s584, s585, s586, s717, s718,
 s719, s720, s721, s722, s723, s724,
 s725, s726, s727, s728, s729, s739,
 s740, s768, s769, s770, s771, s772,
 s773, s774, s776, s777, s778, s779,
 s780, s781, s782, s783, s784, s785,
 s786, s787, s788, s789, s790, s791,
 s792, s793, s794, s853, s854, s855,
 s856, s857, s858, s859, s860, s861,

File Key: a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

s862, s863, s864, s876, s877, s878,
s879, s880, s881, s882, s883, s884,
s885, s886, s887, s888, s889, s890,
s891, s892, s899, s900, s901, s902,
s903, s904, s905, s906, s907, s908,
s909, s910, s911, s912, s913, s914,
s915, s916, s917, s918, s919, s920,
s921, s922, s923, s924, s925, s926,
s927, s928, s929, s930, s931, s932,
s933, s934, s935, s936, s937, s938,
s939, s940, s941, s942, s943, s944,
s945, s946, s947, s948, s949, s950,
s951, s952, s953, s954, s955, s956,
s957, s958, s959, s960, s961, s962,
s963, s964, s965, s966, s967, s968,
s969, s970, s971, s972, s973, s974,
s975, s976, s977, s978, s1079, E368,
E369, E370, E371, E372, E373,
E374, E375, E386, E387, E388,
E389, E390, E391, E392, E393,
E394, E395, E589, E590, E591,
E592, E593, E594, E595, E596, 1104
\DeclareTextSymbolDefault
..... s185, s240, s241, s242,
s243, s244, s245, s246, s247, s248,
s249, s250, s251, s252, s253, s254,
s255, s256, s257, s258, s259, s260,
s261, s262, s263, s264, s265, s266,
s267, s268, s269, s270, s271, s272,
s273, s274, s275, s276, s277, s278,
s280, s281, s291, E114, E116, E118,
E120, E121, E122, E123, E124,
E125, E126, E127, E128, E129,
E130, E131, E132, E133, E134,
E135, E136, E137, E138, E139,
E140, E141, E142, E143, E144,
E145, E146, E147, E148, E149,
E150, E151, E152, E153, E154,
E577, E578, E579, E580, E581,
E582, E583, E584, E597, E598,
E599, E600, E601, E602, E603,
E604, E623, E624, E642, E643,
E644, E645, E646, E647, E648,
E650, E953, E954, E955, E956,
E957, E958, E959, E960, E961,
E962, E963, E964, E965, E966,
E967, E968, E969, E970, E971,
E972, E973, E974, E975, E976,
E977, E978, E979, E980, E981,
E982, E983, E984, E985, E986,
E987, E988, E989, E990, E991,
E992, E993, E994, E995, E996,
E997, E998, E999, E1000, E1001,
E1002, E1003, E1004, E1005, 1103
\DeclareTitlecaseMapping . . . aa576, 1151
\DeclareUnicode 437
\DeclareUnicodeAccent s1042,
s1237, s1238, s1239, s1240, s1241,
s1242, s1243, s1244, s1245, s1246,
s1247, s1248, s1249, s1250, s1251, 1132
\DeclareUnicodeCharacter aa384, 1133
\DeclareUnicodeCommand
..... s1080, s1081, s1082, s1083,
s1164, s1166, s1168, s1215, s1252, 1146
\DeclareUnicodeComposite
..... s1057, s1256, s1257,
s1258, s1259, s1260, s1261, s1262,
s1263, s1264, s1265, s1266, s1267,
s1268, s1269, s1270, s1271, s1272,
s1273, s1274, s1275, s1276, s1277,
s1278, s1279, s1280, s1281, s1282,
s1283, s1284, s1285, s1286, s1287,
s1288, s1289, s1290, s1291, s1292,
s1293, s1294, s1295, s1296, s1297,
s1298, s1299, s1300, s1301, s1302,
s1303, s1304, s1305, s1306, s1307,
s1308, s1309, s1310, s1311, s1312,
s1313, s1314, s1315, s1316, s1317,
s1318, s1319, s1320, s1321, s1322,
s1323, s1324, s1325, s1326, s1327,
s1328, s1329, s1330, s1331, s1332,
s1333, s1334, s1335, s1336, s1337,
s1338, s1339, s1340, s1341, s1342,
s1343, s1344, s1345, s1346, s1347,
s1348, s1349, s1350, s1351, s1352,
s1353, s1354, s1355, s1356, s1357,
s1358, s1359, s1360, s1361, s1362,
s1363, s1364, s1365, s1366, s1367,
s1368, s1369, s1370, s1371, s1372,
s1373, s1374, s1375, s1376, s1377,
s1378, s1379, s1380, s1381, s1382,
s1383, s1384, s1385, s1386, s1387,
s1388, s1389, s1390, s1391, s1392,
s1393, s1394, s1395, s1396, s1397,
s1398, s1399, s1400, s1401, s1402,
s1403, s1404, s1405, s1406, s1407,
s1408, s1409, s1410, s1411, s1412,
s1413, s1414, s1415, s1416, s1417,
s1418, s1419, s1420, s1421, s1422,
s1423, s1424, s1425, s1426, s1427,
s1428, s1429, s1430, s1431, s1432,
s1433, s1434, s1435, s1436, s1437,
s1438, s1439, s1440, s1441, s1442,
s1443, s1444, s1445, s1446, s1447,
s1448, s1449, s1450, s1451, s1452,
s1453, s1454, s1455, s1456, s1457,
s1458, s1459, s1460, s1461, s1462,
s1463, s1464, s1465, s1466, s1467,

File Key: a=ltdirname.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltppageno.dtx,
G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

File Key: a= ltdirchk.dtx, b= ltpplain.dtx, c= ltvers.dtx, d= ltluatex.dtx, e= ltexpl.dtx,
f= ltnode.dtx, g= ltcmd.dtx, h= lthooks.dtx, i= ltcmdhooks.dtx, j= ltalloc.dtx, k= ltcntrl.dtx,
l= lterror.dtx, m= ltpar.dtx, n= ltpara.dtx, o= ltmeta.dtx, p= ltspace.dtx, q= ltlogos.dtx,
r= ltfiles.dtx, s= ltoutenc.dtx, t= ltcounds.dtx, u= ltlengt.dtx, v= ltfssbas.dtx,
w= ltfssaxes.dtx, x= ltfssrc.dtx, y= ltfsscmp.dtx, z= ltfssdcl.dtx, A= ltfssini.dtx,
B= fontdef.dtx, C= preload.dtx, D= ltfntcmd.dtx, E= lttextcomp.dtx, F= ltpageno.dtx,
G= ltxref.dtx, H= ltmiscen.dtx, I= ltmath.dtx, J= ltlists.dtx, K= ltbboxes.dtx, L= lttab.dtx,
M= ltpictur.dtx, N= lthmm.dtx, O= ltssect.dtx, P= ltffloat.dtx, Q= ltidxglo.dtx, R= ltbibl.dtx,
S= ltmarks.dtx, T= ltpage.dtx, U= ltclass.dtx, V= ltkeys.dtx, W= ltflehook.dtx,
X= lthshipout.dtx, Y= ltoutput.dtx, Z= lthyphen.dtx, aa= ltfinal.dtx

\div B383
 \DJ s529, s1138, aa655, 1099
 \dj s539, s1139, aa655, 1099
 \do k3, k7, k16, k26, r66, r69, r72, r135,
 r138, r190, r193, r233, r305, r367,
 r414, r587, r604, r747, r753, a129,
 D90, H438, H459, H567, H577,
 H583, H589, K57, K76, L244, L269,
 M104, M116, M123, M203, M337,
 M339, M362, M365, M394, M396,
 M417, M422, M478, M506, M535,
 M731, M787, P65, P134, R36,
 R65, R82, f69, U232, U249, U480,
 U499, U516, U534, U539, U553,
 U558, U594, U604, U1076, U1113,
 U1195, U1248, U1328, U1417,
 U1694, b13, b14, g1832, a77, a78, 1113
 \DocInput x8, B5, C5, Z4
 \docclearpage 1125
 \document 380
 document (env.) H8
 \document r9, H190, R64, R81, 74
 \documentclass
 d16, x2, B2, C2, U611, U618, U677,
 U680, U890, U985, U1096, Z2, 198
 \DocumentMetadata
 o6, o11, o15, o18, o34, 358
 \documentstyle U616, U1096
 \dollar 1100
 \dospecials a129,
 H438, H459, H567, H577, U1195,
 U1328, U1417, b13, g1833, a77, 1117
 \dot B525
 \doteq B469
 \dotfill f881, f902, b546
 \dots s320, s322, 1107
 \doublehyphendemerits b334
 \doublerulesep L311, L338, L362
 \Downarrow B586
 \downarrow B580
 \downbracefill B538, B557
 \dump aa719

E

\E U1202, U1205, U1233, U1335,
 U1338, U1365, U1424, U1427, U1455
 \edef 1083
 \egroup b450, 1080
 \eject b513
 \ell B311
 \else 1091
 else commands:
 \else: h2195,
 h2332, h2348, h2395, h2430, h2445,

i347, n26, n32, n57, W241, f721,
 g1221, h747, h1002, h1157, h1219,
 h1353, h1360, h1368, h1663, h2168
 \em A550, A577, D42, 589
 \emergencystretch T88, T94, 1109
 \emforce 588
 \emforce A546, A573, A582, 590
 \eminnershape A554, A560, A577, 589
 \emph h2682, D42, 1136
 \empty b448
 \emptyset B318
 \emreset 589
 \emreset A546, A549, A580, 589
 \emrest A549
 \ENC-cmd 1099
 \encodingdefault s990, s1536, s1569,
 s1570, d260, d275, d283, z388,
 A658, A676, A692, A709, B50, 1100
 \EncodingSpecific 1088
 \EncodingSpecificAccent 1088
 \EncodingSpecificAccentedLetter .. 1088
 \EncodingSpecificCommand 1088
 \end l250, x9, B6,
 C6, H197, H204, H206, H234, H260,
 H417, H418, I491, I500, J112, O15,
 O17, f23, U1210, U1214, U1221,
 U1343, U1347, U1353, U1432,
 U1436, U1442, X396, Z5, f719,
 f776, g1296, g1416, g1431, a72, 147
 \endarray L171
 \endcenter H359
 \endcsname 197
 \enddisplaymath I348
 \enddocument ... H8, H73, H75, X352, 1141
 \endenumerate J240
 \endeqnarray I380, I427
 \endequation I351
 \endfilecontents U1100
 \endflushleft H409
 \endflushright H411
 \endgraf n133, n171, K94, b445, 1118
 \endgroup 1121
 \EndIncludeInRelease
 h2182, h2200, h2205,
 h2215, h2219, h2222, h2248, h2261,
 h2271, h2288, h2300, h2312, h2322,
 h2352, h2371, h2434, h2438, b579,
 h2520, h2523, h2632, h2638, h2645,
 h2650, h2657, h2661, h2668, h2672,
 h2705, h2710, b613, b621, i31, i45,
 b646, i228, i295, i420, i439, i491,
 i495, b686, b691, i547, i574, i577,
 i589, b701, b706, l157, l161, l204,
 l210, n37, n68, c71, p21, p31, p65,

File Key: a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

p78, p86, p91, p104, p110, p152, p166, p178, p189, p206, p218, p252, p269, p307, p329, p366, p399, p432, p436, p446, p452, p466, p473, r83, r140, r195, r256, r277, r289, r352, r356, r435, r473, r496, r531, r552, r571, r578, r597, r614, r627, r631, r649, r660, r670, r709, r736, s103, s123, s168, s175, s330, s352, s367, s375, t28, t33, t68, t73, t98, t125, t134, t176, t182, t202, t205, u10, u14, v52, v73, v231, v248, v294, v304, v314, v366, v375, v455, v460, v495, v500, v517, v535, v574, v607, v723, v735, d234, w384, w390, w403, w415, w422, d257, w505, w520, w599, w611, w622, w629, w636, d280, w709, d284, w727, w734, w738, w741, x157, x160, x176, x549, x555, y21, y143, z77, z105, z179, z209, z239, z271, z306, z349, z410, z416, z421, z504, z511, z556, z563, z837, z879, z892, z899, z1087, z1095, A67, A100, A122, A227, A248, A300, A347, A376, A387, A434, A468, A494, A530, A539, A576, A588, A595, A629, A635, A670, A686, A703, A718, B80, B90, B109, B118, B633, B640, D33, D40, G28, G45, G61, G75, G82, G110, G127, G137, G149, G160, G171, H71, H110, H118, H124, H157, H170, H232, H282, H301, H315, H324, H334, H341, H351, H356, H380, H396, H405, H443, H464, H495, H505, H523, H538, H550, H572, H580, I86, I95, I117, I124, I143, I148, I156, I160, I175, I183, I232, I249, I279, I287, I316, I343, I407, I416, I448, I453, I473, I485, I494, I503, J132, J137, K13, K22, K69, K86, K101, K113, K124, K131, K188, K195, K243, K251, K302, K320, K397, K413, K430, K439, K444, K467, K474, e10, L64, L69, e18, L156, L164, e27, L226, L231, M15, M19, M38, M47, M68, M76, M88, M96, M111, M121, M150, M155, M171, M182, M249, M263, M368, M425, M463, M469, M500, e110, M530, M555, M571, M582, M595, M603, M624, M641, M651, M655, e131, M745, M800, M819, M836, e144, O19, O29, e153, O167, O173, O178, O192, O198, O224, O246, e165, P104, e173, P172, P231, P246, P293, P306, P351, e188, P368, P411, e194, P417, P426, P430, P439, P445, P449, P481, P498, P516, f12, P558, P564, f18, R24, R32, R76, R92, T47, T68, T75, U21, U26, f60, U49, f64, U61, U84, U93, U99, U111, U142, U149, U174, U181, U199, U210, U244, U261, U272, U281, U298, U310, U331, U344, U357, U367, U403, U419, U428, U460, U470, U510, U526, U548, U563, U577, U584, U599, U608, U642, U649, U666, U673, U752, U779, U807, U960, U1017, U1047, U1054, U1236, U1367, U1457, f224, f229, W14, W23, W88, W140, W179, W191, W200, W245, W256, W263, W299, W322, W343, W357, W362, W385, W405, W430, W475, W495, W502, W532, W537, f324, f352, X429, X466, X483, X487, f380, Y53, Y62, f385, Y180, Y198, f404, Y367, Y372, f418, Y420, Y466, Y654, Y712, Y815, Y834, Y897, Y915, Y957, Y978, f459, f469, a309, Y1220, Y1389, Y1471, f501, Y1565, f510, Y1687, Y1814, Y1924, f537, Y1932, Y1966, f543, Y1995, a319, Y2215, f557, Y2233, f562, Y2280, Y2324, aa15, aa19, aa29, aa33, aa51, aa69, aa79, aa86, aa94, aa141, aa146, aa198, aa222, aa292, aa297, aa338, aa344, a25, aa440, aa487, f663, f678, f729, f733, f765, f774, f824, f831, f853, f864, f867, f895, f916, b87, b101, b118, b123, b133, g1072, g1075, b137, g1090, g1103, g1106, b147, b150, g1224, g1227, g1255, g1258, g1267, g1275, g1278, b167, g1401, g1405, b181, b185, b219, b224, b232, b240, a53, b288, b300, g2865, g2872, b359, b367, h85, h101, h129, h145, h157, h169, h185, h198, h218, h228, h248, h263, h281, h287, h307, h325, h329, h575, h622, h639, h643, h665, h682, h710, h766, h794, h822, h845, h848, h868, h871, h884, h902, h907, h910, h916, h921, h924, h959, h992, h1041, h1048, h1064, h1071, h1091, h1095, h1146, h1150, h1166, h1170,

File Key: a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspac.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

h1198, h1201, h1224, h1228, h1245,	
h1250, h1257, h1261, b471, h1297,	
h1321, h1397, h1422, h1454, b488,	
h1486, h1574, h1648, h1877, h1972,	
h1977, h1982, h2053, h2059, h2083,	
h2099, h2106, h2111, h2133, h2153, 29	
\EndIncludeRelease	c78
\endinput	1093
\enditemize	J251
\endline	I188, b445
\endlinechar	a95, a96, a97, r684, r718, a207, f37, f39, f44, U372, U373, U374, aa329, b380, 676
\endlist	J98, J240, J251
\endlrbox	K155
\endmath	I346
\endminipage	K363
\EndModuleRelease	
	h2827, i613, n176, o36, c111, c112, c152, E809, S386, g2897
\endpicture	M49
\endscname	1079
\endsloppypar	T92
\endtabbing	L84
\endtabular	L171
\endtabular*	L171
\endtrivlist	H359, H409, H411, H467, I515, J100, J101, L85, N39
\endverbatim	H466, H493
\enlargethispage	Y1859, 1127
\enlargethispage*	Y1859
\enskip	p475
\enspace	p463, p471
\ensuremath	t178, I429, P409, P416, P437, P444, 1116
enumerate (env.)	J231
\enumerate	J231
environments:	
array	L168
center	H358
displaymath	I345
document	H8
enumerate	J231
eqnarray	I357, I516
eqnarray*	I425
equation	I349, I504
filecontents	875, U1100
flushleft	H408
flushright	H410
itemize	J242
list	J34
lrbox	725
math	I345
minipage	726
\EveryShipout	963

File Key: a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

```

\ExecuteOptions x57, x70, E843, E880, U587
\exhyphenpenalty ..... b322, b507
\exists ..... B324
\exp ..... I31
exp commands:
  \exp:w ..... g2260, h737, h789, 241
  \exp_after:wN ..... h2325,
    h2328, h2392, h2856, i170, i245,
    i299, i300, i353, i471, n65, z319,
    z322, T41, W73, W236, W237,
    g258, g274, g328, g334, g348,
    g1064, g1084, g1088, g1099, g1101,
    g1138, g1187, g1217, g1247, g1254,
    g1293, g1427, g1487, g1511, g1527,
    g1545, g1551, g1555, g1586, g1620,
    g1627, g1640, g1646, g1652, g1658,
    g1664, g1670, g1676, g1682, g1713,
    g1841, g2263, g2264, g2372, g2373,
    g2374, g2375, g2376, g2377, g2378,
    g2408, g2477, h56, h61, h378,
    h379, h424, h736, h788, h1001,
    h1003, h1142, h1307, h1468, h1662,
    h1771, h1802, h1887, h2019, h2147, 241
  \exp_args:Nv ..... h1208, h1916, h1923, h2017
  \exp_args:Nx ..... X29,
    g2339, g2342, g2365, h430, h472, h542
  \exp_args_generate:n ..... aa598
  \exp_end: ..... g2263, h737, h789
  \exp_last_unbraced:Ne ..... h2849,
    g2410, h1026, h1059, h1083, h1175
  \exp_last_unbraced:Nf ..... i110, h1188, h1826
  \exp_last_unbraced:NNf ..... h1117
  \exp_last_unbraced:NNNo ..... i116, i474, h381, h1975
  \exp_last_unbraced:NnNo ..... g2122
  \exp_last_unbraced:NNo ..... i364
  \exp_not:N ..... i108, i110, i111,
    i112, i184, i199, i206, i207, i214,
    i219, i259, i274, i281, i282, i286,
    i290, i307, i352, i358, i376, i385,
    i459, i460, i480, i485, i513, i540,
    i541, i568, i569, n63, n64, n72, n73,
    V148, V149, V150, X31, X95, X125,
    aa605, g90, g118, g119, g151, g153,
    g154, g160, g162, g203, g215, g232,
    g233, g234, g236, g304, g849, g859,
    g929, g942, g966, g984, g991, g1029,
    g1036, g1037, g1069, g1111, g1112,
    g1113, g1117, g1120, g1121, g1123,
    g1142, g1145, g1146, g1147, g1155,
    g1202, g1221, g1235, g1241, g1242,
    g1306, g1307, g1687, g1688, g1796,
    g1798, g1799, g1863, g2401, g2404,
    g2405, g2406, g2408, h389, h1111,
    h1115, h1133, h1136, h1137, h1139,
    h1183, h1436, h1437, h1564, h1566, 310
  \exp_not:n ..... h2847, i171, i184,
    i187, i189, i224, i246, i259, i262,
    i264, i292, i356, i386, i388, i389,
    i391, i392, S149, S150, S166, S167,
    S168, S241, S242, S243, S245, S246,
    S247, S249, S250, S251, S253, S254,
    S255, S257, S258, S259, S261, S262,
    S263, V132, W211, W225, X30,
    g113, g117, g121, g124, g128, g145,
    g152, g164, g216, g235, g304, g307,
    g308, g561, g695, g796, g807, g809,
    g810, g878, g930, g943, g954, g966,
    g983, g1068, g1070, g1111, g1124,
    g1156, g1191, g1411, g1413, g1426,
    g1496, g1497, g1540, g1600, g1797,
    g1913, g1914, g1954, g1955, g2137,
    g2146, g2163, g2177, g2189, g2401,
    h114, h241, h545, h1008, h1032,
    h1130, h1134, h1140, h1141, h1570, 310
  \exp_stop_f: .....

```

File Key: a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr1.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspc.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

... g1338, g1900, h1209, h1350, h1358
 expandable commands:
 \expandable_grab_{type}:w 146
 \expandafter 681
 expandafter commands:
 \expandafter: D88, L269
 \ExpandArgs 79
 \ExpandArgs
 .. e174, e190, e193, f534, f535, aa602
 \expanded U820, aa111, 232
 \ExplSyntaxOff h2863, i381, i614, n177,
 o37, r529, z304, z347, e143, e163,
 e186, S387, T45, V213, W12, W86,
 W138, W243, W254, W266, W403,
 W428, W473, W530, W585, X427,
 X474, X514, aa575, aa653, g2898, 881
 \ExplSyntaxOn
 .. i3, i381, n3, o2, r518, z277, z310,
 e135, e158, e177, S3, T22, V3, W7,
 W29, W92, W207, W251, W391,
 W411, W436, W509, W578, X5,
 X472, X509, aa565, aa576, g8, h3, 161
 \extracolsep L167
 \extraffloats b152, b189, b274

F

 \fam d22, d26, d38, v15, b98, b375
 \family 1082
 \familydefault s1570, z389, A444,
 A659, A677, A693, A710, B120, 578
 \fbox 725
 \fbox K173, K186, K193, 1097
 \fboxrule K171, K207,
 K210, K216, K218, K225, K226, aa151
 \fboxsep K171,
 K177, K206, K211, K221, K223, aa150
 \fi 1080
 fi commands:
 \fi: h2197, h2329,
 h2334, h2350, h2397, h2432, h2447,
 i342, i344, i345, i349, i354, i361,
 n26, n33, n58, n96, W241, g807,
 g810, g877, g1221, g1340, g2260,
 g2264, h749, h1004, h1159, h1182,
 h1222, h1354, h1362, h1368, h1665,
 h2020, h2123, h2131, h2148, h2170, 314
 \filbreak b511
 file commands:
 \g_file_curr_name_str h2696
 \file_full_name:n r525, W35, 942
 \file_parse_full_name_apply:nN ..
 .. W32, W47, W90, W93, W95
 \l_file_search_path_seq 949

file internal commands:
 __file_parse_full_name_area:nw ..
 W102, W105, W109
 __file_parse_full_name_auxi:nN ..
 W97, W100
 __file_parse_full_name_base:nw ..
 W108, W112, W124
 __file_parse_full_name_tidy:nnnN ..
 W119, W120, W122, W127
 file/.../after 933
 file/.../before 933
 file/after 933
 file/before 933
 filecontents (env.) 875, U1100
 \filecontents U1100
 filehook internal commands:
 __filehook_clear_replacement_-
 flag: W418, W421, W521
 __filehook_drop_extension:N ..
 W44, W49, W523
 __filehook_drop_extension_-
 aux:nnn W54, W57
 __filehook_file_name_compose:nnn
 .. W218, W407, W415, W416, W426
 __filehook_file_parse_full_-
 name:nN W28, W30, W30,
 W53, W238, W395, W413, W415, 939
 __filehook_file_pop:
 W59, W70, W527
 __filehook_file_pop_assign:nnnn
 W59, W73, W79, W529
 __filehook_file_push:
 W59, W62, W525
 __filehook_file_subst_begin:nnn
 W413, W423, W423, 949
 __filehook_file_subst_cycle_-
 error:NN
 W458, W463, W463, W468, W470, 951
 __filehook_file_subst_loop:NN ..
 W432, W445, W453, W462, 951
 __filehook_file_subst_tortoise_-
 hare:nn
 W425, W432, W435, W437, W460, 950
 __filehook_full_name:nn
 W30, W34, W38
 __filehook_if_file_replaced:TF ..
 W418, W419, W519, 950
 __filehook_if_no_extension:nTF ..
 W44, W44, W511
 \g__filehook_input_file_seq W59, 940
 \l_filehook_internal_tl W59
 __filehook_log_file_record:n ...
 W548, W548, W562, W564

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfnctcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

```

\g__filehook_nesting_level_int ...
    .. W545, W550, W553, W554, W560
\__filehook_normalize_file_-
    name:w ..... W407, W414, W517
\__filehook_resolve_file_subst:w
    ..... W407, W410, W412, W515
\__filehook_set_curr_file:nNN ...
    ..... W393, W513
\__filehook_set_curr_file_-
    assign:nnnNN ..... W396, W398
\__filehook_subst_add:nn ...
    .. W206, W208, W208, W252, 944
\__filehook_subst_empty_name_-
    chk:NN ..... W208, W236, W240
\__filehook_subst_file_normalize:Nn
    .. W208, W216, W218, W230, W234
\__filehook_subst_remove:n ...
    ..... W208, W222, W253
\filename@parse ..... 6, 1
\filesize ..... e72, e118
\fill ..... p456
\finalhyphendemerits ...
    .. H367, H371, H377, b335, 1138
\finalstrut ..... 1106
\FirstMark ..... S274, S377, 849
\firstmark . T80, Y651, Y710, Y2255, 847
flag internal commands:
    \flagU__filehook_file_replaced . 950
    \flagU__filehook_file_replaced . W418
flag commands:
    \flag_clear:n ..... W422
    \flag_if_raised:nTF ... W420, W441
    \flag_new:n ..... W418
    \flag_raise:n ..... W442
\flat ..... B328
\floatingpenalty P469, P488, P506, b332
\floatactionfraction ..... P278, Y2336
\floatasep ..... Y730,
    Y748, Y755, Y2142, Y2192, Y2341
\flushbottom ..... T84
flushleft (env.) ..... H408
\flushleft ..... H408
flushright (env.) ..... H410
\flushright ..... H410
\fmtname ..... c1, c41, c43,
    c46, c48, c51, c60, U686, U690, 1085
\fmtversion l2, c1, c19, c41, c43, c46, c48,
    c51, c60, c105, c118, c166, s1538,
    d276, A225, L1, M1, U169, U177,
    U703, U706, Y4, aa662, aa688, 1085
\fnsymbol ..... 449
\fnsymbol ..... t141
\font ..... s297, s298,
    s299, s439, s446, s799, s806, s866,
    s1003, s1004, s1060, s1165, s1167,
    s1169, s1216, v81, v87, v89, x84,
    A553, A586, A592, A618, C8, C9,
    C10, D85, E6, E612, E626, E633,
    H461, f842, f845, f857, f860, b539, b544
\fondimensions ..... s297, s298,
    s299, s439, s446, s799, s806, A553,
    A586, A592, D85, E6, E612, E626,
    E633, M126, M129, M679, b539, b544
\fontencoding ..... s867, s1569, v251,
    v286, v298, v308, d259, d260, d282,
    d283, z388, A658, A676, A692, B16,
    B24, B67, B74, B83, B85, E36, 468
\fontfamily . v279, A6, A9, A12, A145,
    A156, A482, A485, A488, A738,
    B58, B69, B76, B87, E28, E30, E32,
    E34, E85, E87, E89, E91, E916, 468
\fontname ..... s1004, v89
\fontseries ..... v279, w393,
    w394, w405, w407, w417, w419,
    A15, A18, A258, A259, A260, A261,
    A281, A282, A283, A284, A319,
    A320, A321, A322, A339, A340,
    A341, A342, A355, A356, A357,
    A358, A366, A367, A368, A369,
    A382, A385, A476, A479, A739, 1137
\fontseriesforce . w398, w409, w420, 496
\fontshape ..... s449,
    s809, v279, w528, w533, w538,
    w614, w615, w624, w626, w631,
    w633, w684, w688, w691, w694,
    w697, w700, w703, w706, w713,
    A21, A24, A27, A30, A740, E636, 512
\fontshapeforce . w618, w627, w634, w714
\fontsize ..... q6,
    s302, s328, s360, s868, s1218, s1254,
    v79, v318, A641, A741, E104, E672,
    E933, P409, P416, P437, P444, 1098
\fontsubfuzz .. x441, x475, H49, H89, 1094
\fO0 ..... 81
\foo ..... 308
\footins . P390, P465, P484, P502,
    Y316, Y317, Y318, Y319, Y377,
    Y424, Y486, Y494, Y498, Y521, 1111
\footnote ..... P452, 1081
\footnotemark ..... O10, P518, 1105
\footnoterule ... K369, P394, Y497, 1111
\footnotesep K393, K410, K427, P451,
    P468, P477, P487, P495, P505, P513
\footnotesize ..... K385, K403, K420, P466, P485, P503
\footnotetext ..... O12, P535, 1081
\footref ..... P547, 1142
\footskip ..... Y77, Y641, Y700

```

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=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\forall	B323
fp commands:	
\fp_eval:n	e159
\fp_eval	78
\fp_eval	e155, e167, e169, 78
\frac	I354, 1087
\frame	K157, K233
\framebox	725
\framebox	K180, 1097
\frenchspacing	r46, r116, r174, H466, H492, H582, f882, f903, b431
\frown	B451
\fussy	T93
\futurelet	p411, p419, D83, I256, e49, L359, f782, f796
G	
\g_@@_\meta_{hook}_code_prop	221
\g_@@_\meta_{hook}_declared_tl	221
\g_@@_\meta_{hook}_parameter_tl	221
\g_@@_\meta_{hook}_reversed_tl	221
\Gamma	B297
\gamma	B270
\gcd	133
\gdef	1135
gdef commands:	
\gdef_	I269
\ge	B417, B419
\Generic*	1105
\GenericError	I18, I85, I111, I164, x62, 1101
\GenericInfo	I4, I104, I130, I182, c106, c108, c119, c122, c127, c162, c163, c168, c172, c176, c197, x31, x34, x39, x75, e63, U66, U74, U104, U107, U1620, 1101
\GenericWarning	I11, I94, I120, I141, I149, I173, I195, x42, x47, x50, x78, 1101
\geq	B415, B417
\getanddefinefonts	1079
\GetDocumentCommandArgSpec	g2879
\GetDocumentEnvironmentArgSpec	g2879
\GetFileInfo	B3
\gets	B439, B441
\gg	B433
\glossary	1135
\global	1093
\globaldefs	v618, x231, z60, z89, z160, z191, z221, z253, b372
\glossary	839
\glossary	O182, Q23, Q35, S163, T25, T36, T53, T61, Y625, Y684, 848
\glossaryentry	Q32
\goodbreak	f883, f904, b511
\grave	B517
H	
\H	124, s230, s385, s469, s601, s609, s628, s636, s753, s1246, s1385, s1386, s1413, s1414, E177, E201
\halign	I127, I210, I371, I532, b534, 1079
\hang	1110
\hangafter	b374
\hangindent	O139, b405
\hat	B523
\hbadness	v672, v679, v714, v733, X238, X240, X290, X292, b318
\hbar	B335
\hbox	1126
\hbox commands:	
\hbox:n	X382
\hbox_set:Nn X170, X217, X243, X268, X297
\hbox_set_to_wd:Nnn	X241, X293, X329
\hbox_unpack:N	S291, S322, X251, X295
\hbox_unpack_drop:N X180
\hbox_to_ou	1095
\headheight	Y75, Y630, Y689
\headsep	Y76, Y639, Y698
\heartsuit	B333

File Key: a=ltdirname.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrn.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfssrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfnctmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\height s872, s1222, K31, K34, 1086
 \hfil 711
 \hfill 1082
 \hfuzz v680, T89, T90, T96,
 T97, X237, X239, X289, X291, b390
 \hglue b501
 \hideoutput b692
 \hideskip b309, b525
 \hidewidth s327, s329,
 s358, s362, s390, s391, s394, s397,
 s476, s477, s481, s484, s486, s489,
 s501, s506, s522, s760, s761, s764,
 s767, s834, s837, s1253, s1255, b525
 \hline L358, L361, 1083
 \hoffset b406
 \holdinginserts b339
 \hom I29
 hook commands:
 \hook_activate_generic:n
 ... h2642, h2736, h2755, h2768,
 h288, h288, h290, h308, h326, h328, 230
 \hook_debug_off: ... h2714, h7, h13, 206
 \hook_debug_on: ... h2713, h7, h8, 206
 \hook_disable:n h2724, h2724
 \hook_disable_generic:n
 ... h2644, h2729, h2761,
 h264, h264, h266, h282, h285, 204
 \hook_gclear_next_code:n
 ... h2674, h2100, h2100, 205
 \hook_gput_code:nnn h2654, h483,
 h483, h485, h576, h578, h671, h688, 204
 \hook_gput_code_with_args:nnn ...
 ... h2656, h483, h491, h621, 204
 \hook_gput_next_code:nn
 ... h2665, h678, h693,
 h2039, h2039, h2041, h2054, h2056, 205
 \hook_gput_next_code_with_-
 args:nn . h2667, h2047, h2058, 1151
 \hook_gremove_code:nn
 ... h2676, h925, h925, h927, h960, 205
 \hook_gset_rule:nnnn h2716,
 h2718, h2721, h1262, h1262, 205
 \hook_if_empty:n
 ... h2336, h2338, h2353, h2355
 \hook_if_empty:nTF h2336,
 h2722, X63, X164, h1793, h1896, 197
 \hook_if_empty_p:n h2336,
 X67, X113, X377, h1846, h1941, 205
 \hook_log:n . h2712, h1749, h1749, 206
 \hook_new:n ... h2619, S165, X153,
 X154, X155, X156, X157, X158,
 X159, h67, h69, h88, h223, h833, 203
 \hook_new_pair:nn
 ... h2623, n6, n7, h199, h201, h221, 203
 \hook_new_pair_with_args:nn ... 203
 \hook_new_pair_with_args:nnn ...
 ... h2631,
 h199, h199, h203, h219, h226, 203
 \hook_new_reversed:n
 ... h2621, h170, h172, h188, h224, 203
 \hook_new_reversed_with_args:nn .
 ... h2629,
 h170, h170, h174, h186, h197, 1151
 \hook_new_with_args:nn
 ... h2627, h67, h67, h71, h86, h100, 203
 \g_hook_patch_action_list_tl ...
 ... i6, i91, i124
 \hook_provide:n h2724, h2731
 \hook_provide_pair:nn . h2724, h2745
 \hook_provide_reversed:n h2724, h2738
 \hook_show:n h2711, h1749, h1754, 206
 \hook_use:n
 ... h2210, h2701, n22, n31, n48,
 n56, n91, n98, S140, X62, X65, X71,
 X75, X123, h1415, h2112, h2112,
 h2114, h2134, h2136, h2154, h2156, 283
 \hook_use:nnw
 ... h2181, h2183, h2183, h2185,
 h2201, h2203, h2211, h2703, h2152, 204
 \hook_use_once:n
 ... h2249, h2251, h2264, h2702, 281
 \hook_use_once:nnw h2249,
 h2249, h2256, h2262, h2269, h2704, 204
 hook internal commands:
 \c__hook_ h1254, h1258
 \g__hook_??_code_prop h1251
 \c__hook_??_parameter_tl h1251
 \g__hook_??_reversed_tl h1251
 \g__hook_{hook}_code_prop 221
 \g__hook_{hook}_labels_clist ... 225
 \g__hook_{hook}_reversed_tl ... 222
 __hook_activate_generic:n ... h288
 __hook_activate_generic:nn
 ... h2753, h291, h292, h310
 __hook_activate_generic_pair:nn
 ... h2724, h2750, h2754, h2782
 __hook_activate_generic_-
 reversed:n
 ... h2724, h2743, h2752, h2755, h2775
 \g__hook_all_seq
 ... h2854, h28, h108, h137, h1378, h1404
 __hook_apply_-rule_->:nnn .. h1724
 __hook_apply_-rule_-<:nnn .. h1724
 __hook_apply_-rule_<:nnn ... h1724
 __hook_apply_-rule_>:nnn ... h1724
 __hook_apply_-rule_xE:nnn .. h1724
 __hook_apply_-rule_xW:nnn .. h1724

File Key: a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfnctcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

```

\__hook_apply_label_pair:nnn ...
    ..... h1514,
    h1515, h1595, h1596, h1651, h1651, 263
\__hook_apply_rule:nnn .....
    ..... h1661, h1667, h1667, 267
\__hook_apply_rule:nnnN .....
\__hook_apply_rule_->:nnn ... h1702
\__hook_apply_rule_<-:nnn ... h1702
\__hook_apply_rule_<:nnn ... h1674
\__hook_apply_rule_>:nnn ... h1674
\__hook_apply_rule_xE:nnn ... h1688
\__hook_apply_rule_xW:nnn ... h1688
\__hook_braced_cs_parameter:n ...
    ..... h2850, h1027, h1084, h1109,
    h1119, h1171, h1171, h1173, h1199
\__hook_braced_hidden_loop:w ...
    ..... h1171, h1175, h1178, h1184
\__hook_braced_parameter:n . i221,
    i510, h1202, h1202, h1204, h1225,
    h1227, h1436, h1437, h1564, h1566
\__hook_braced_real_loop:w .. h1202
\__hook_chk_args_allowed:nn ...
    ..... h499, h623,
    h623, h625, h640, h642, h2076, 277
\__hook_clear_next:n .....
    ..... h2080, h2096, h2101,
    h2102, h2102, h2104, h2107, h2109, 274
\__hook_clist_gput:Nn .....
    ..... h1443, h1445, h1475,
    h1477, h1538, h1619, h1649, h1650
\__hook_cmd_begindocument_code: .
    ..... i46, i54, i59, i62, i611, 321
\__hook_cmd_if_scannable:Nn ... i450
\__hook_cmd_if_scannable:NnTF ...
    ..... i408, i427, i450, 316
\__hook_cmd_patch_xparse:Nnn ...
    ..... i128, i149, i149
\__hook_cmd_try_patch:nn i52, i63, i63
\__hook_code_gset:nn .....
    ..... h2307, h118, h1049, h1049,
    h1051, h1063, h1065, h1067, h1434
\__hook_code_gset_aux:nnn ...
    ..... h1012, h1049,
    h1052, h1054, h1056, h1057, h1070
\__hook_code_gset_auxi:nnnn h993,
    h1013, h1038, h1040, h1047, h1078, 248
\__hook_cs_end:w .....
    .. h1171, h1190, h1191, h1192, h1197
\__hook_cs_gput_right:nnn .....
    ..... h2279, h537,
    h993, h993, h995, h1042, h1044, h2081
\__hook_cs_gput_right_fast:nnn ..
    ..... h993, h1001, h1007, h1045
\__hook_cs_gput_right_slow:nnn ...
    ..... h993, h1003, h1009, h1046
\__hook_cs_if_empty:N .....
    ..... h1151, h1153, h1167, h1169
\__hook_cs_if_empty:NTF .....
    ..... h2344, h2345,
    h1151, h1816, h1823, h2077, h2079
\__hook_cs_if_empty_p:N ..... h1151
\__hook_cs_parameter_count:N ...
    ..... h1171, h1176, h1186
\__hook_cs_parameter_count:w ...
    ..... h1171, h1188, h1195, h1196
\l__hook_cur_hook_t1 .....
    ..... h29, h1500, h1581, h1708, h1719, 269
\__hook_curr_name_pop: .....
    ..... h2692, h2796, h415, h447, 234
\__hook_curr_name_push:n .....
    ..... h2690, h2695, h415, h429, 292
\__hook_curr_name_push_aux:n ...
    ..... h415, h430, h431
\__hook_currname_or_default: ...
    ..... h2534, h2578, h333, h341,
    h345, h361, h362, h362, h534, h605, 231
\__hook_debug:n .....
    ..... i19, i28,
    i39, i48, i49, i65, i69, h7, h7, h20,
    h528, h599, h1377, h1386, h1403,
    h1406, h1427, h1449, h1459, h1481,
    h1519, h1539, h1600, h1620, h1676,
    h1683, h1690, h1698, h1704, h1715, 218
\g__hook_debug_bool h6, h10, h15, h21
\__hook_debug_gset: h7, h11, h16, h18
\__hook_debug_label_data:N h1519,
    h1560, h1600, h1641, h1737, h1737
\__hook_debug_print_rules:n .....
    ..... h2023, h2023
\__hook_declare_DEPRECATED_-
    generic:NNn ..... h736, h757, h788
\__hook_declare_DEPRECATED_-
    generic:NNw ..... h752, h758, h759
\__hook_def_cmd:w .....
    ..... i13,
    i14, i192, i198, i267, i273, i299, 312
\g__hook_delayed_patches_prop ...
    ..... i17, i51, i57, i58
\__hook_DEPRECATED_generic_-
    warn:Nn ..... h735, h742, h787, h953,
    h986, h1273, h1304, h1769, h1885, 242
\__hook_DEPRECATED_generic_-
    warn:Nn ..... h742
\__hook_DEPRECATED_generic_-
    warn:Nw ..... h742
\__hook_DEPRECATED_generic_-
    warn:W ..... h743, h744
\__hook_DEPRECATED_warn:nn .....
    ..... h2726, h2733, h2740, h2747, h2758,

```

File Key: a=ltDIRchk.dtx, b=ltPLAIN.dtx, c=ltVERS.dtx, d=ltLUATEX.dtx, e=ltEXPL.dtx,
f=ltDEFNS.dtx, g=ltCMD.dtx, h=ltHOOKS.dtx, i=ltCMDBOOKS.dtx, j=ltALLOC.dtx, k=ltCNTRL.dtx,
l=ltERROR.dtx, m=ltPAR.dtx, n=ltPARA.dtx, o=ltMETA.dtx, p=ltSPACE.dtx, q=ltLOGOS.dtx,
r=ltFILES.dtx, s=ltOUTENC.dtx, t=ltCOUNTS.dtx, u=ltLENGTH.dtx, v=ltFSSBAS.dtx,
w=ltFSSAXES.dtx, x=ltFSSTRC.dtx, y=ltFSSCMP.dtx, z=ltFSSDCL.dtx, A=ltFSSINI.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltFNTCMD.dtx, E=ltTEXTCOMP.dtx, F=ltPAGENO.dtx,
G=ltXRREF.dtx, H=ltMISCEN.dtx, I=ltMATH.dtx, J=ltLISTS.dtx, K=ltBOXES.dtx, L=ltTAB.dtx,
M=ltPICTUR.dtx, N=ltTHM.dtx, O=ltSECT.dtx, P=ltFLOAT.dtx, Q=ltIDXGLO.dtx, R=ltBIBL.dtx,
S=ltMARKS.dtx, T=ltPAGE.dtx, U=ltCLASS.dtx, V=ltKEYS.dtx, W=ltFILEHOOK.dtx,
X=ltSHIPOUT.dtx, Y=ltOUTPUT.dtx, Z=ltHYPHEN.dtx, aa=ltFINAL.dtx

h2765, h2772, h2779, h2784, h2784
_hook_disable:n
..... i209, i506, h264, h267, h268
_hook_do_deprecated_generic:Nn
..... h752, h752,
h954, h987, h1274, h1305, h1770, h1886
_hook_do_deprecated_generic:Nw
..... h752, h753, h754
_hook_double_hashes:n
.... i188, i263, i325, i325, i363,
h546, h1024, h1033, h1081, h1137, 253
_hook_double_hashes:w
.. i325, i326, i327, i359, i363, i366, 313
_hook_double_hashes_group:n ...
..... i325, i333, i362
_hook_double_hashes_output:N ..
..... i325, i330, i338, 313
_hook_double_hashes_space:w ...
..... i325, i334, i365
_hook_double_hashes_stop:w ...
..... i325, i341, i361
\c_hook_empty_tl h35, h1243
_hook_end_document_label_-
check: h415, h454, h455, h462
_hook_exp_not:n i171, i172,
i176, i187, i246, i247, i251, i262, 309
_hook_exp_not:NN .. i13, i13, i193,
i199, i219, i224, i268, i274, i290, i292
_hook_file_hook_normalize:n ...
..... h2236,
h700, h869, h869, h872, h874, 244
\l_hook_front_tl .. h1489, h1530,
h1533, h1536, h1538, h1539, h1540,
h1553, h1554, h1611, h1614, h1617,
h1619, h1620, h1621, h1634, h1635
\c_hook_generic_<type>/./<place>_tl
..... 241
\c_hook_generic_class/./after_-
tl h885
\c_hook_generic_class/./before_-
tl h885
\c_hook_generic_cmd/./after_tl h885
\c_hook_generic_cmd/./before_tl
..... h885
\c_hook_generic_env/./after_tl h885
\c_hook_generic_env/./before_tl
..... h885
\c_hook_generic_env/./begin_tl h885
\c_hook_generic_env/./end_tl .. h885
\c_hook_generic_file/./after_tl
..... h885
\c_hook_generic_file/./before_-
tl h885
\c_hook_generic_include/./after_-
tl h885
\c_hook_generic_include/./before_-
tl h885
\c_hook_generic_include/./end_-
tl h885
\c_hook_generic_package/./after_-
tl h885
\c_hook_generic_package/./before_-
tl h885
_hook_generic_parameter:n ...
..... h1019, h1238, h1249
_hook_generic_parameter:w ...
..... h1239, h1240
\c_hook_generics_file_prop ...
..... h861, h908
\c_hook_generics_prop
..... h803, h831, h885, h903, h905
\c_hook_generics_reversed_ii_-
prop h811, h834, h908
\c_hook_generics_reversed_iii_-
prop h814, h837, h908
_hook_gput_code:nnn h483,
h488, h494, h497, h579, h580, h654
_hook_gput_code_store:nnn
..... h483, h510, h513
_hook_gput_next_code:nn .. h661,
h2044, h2050, h2057, h2060, h2060
_hook_gput_next_do:nn
..... h662, h679, h693, h2066,
h2071, h2071, h2073, h2084, h2086, 239
_hook_gput_next_do:Nnn h2088, h2091
_hook_gput_undeclared_hook:nnn
... h644, h644, h655, h672, h688, 239
_hook_gremove_code:nn
.. h925, h928, h929, h954, h962, h987
_hook_gset_rule:nnnn
..... h1262, h1264, h1267,
h1269, h1274, h1298, h1300, h1305
_hook_guess_arg_count:NN
.... i467, i467, i469, i492, i494, i505
_hook_guess_arg_count:nw
..... i467, i480, i485, i488
_hook_guess_arg_count:wN
..... i467, i471, i475
\c_hook_hash_t1
i11, i254, i255, i257, i344, h1183, 252
\c_hook_hashes_t1
i11, i179, i180, i182, i345, h1111, 313
\g_hook_hook_curr_name_t1
.... h32, h364, h374, h415, h427,
h442, h443, h450, h460, h461, h481, 234

File Key: a=ltdefns.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltcmd.dtx, g=ltcmdhooks.dtx, h=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltospace.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmiscren.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=lthyphen.dtx, aa=ltfinal.dtx

```

\__hook_hook_gput_code_do:n .. .
    ..... h240, h257,
    h483, h517, h526, h587, h597, h647
\__hook_if_cmd_hook:n .. .
    ..... h2419, h2421, h2435
\__hook_if_cmd_hook:nTF .. .
    ..... h2419, h2437, h1988
\__hook_if_cmd_hook:w .. h2422, h2423
\__hook_if_cmd_hook:wTF .. . h2419
\__hook_if_cmd_hook_p:n .. . h2419
\__hook_if_cmd_hook_p:w .. . h2419
\__hook_if_declared:n .. . h2384
\__hook_if_declared:nTF .. .
    ..... h2384, i67, h75,
    h93, h178, h207, h210, h297, h315,
    h629, h833, h998, h1015, h1987, 223
\__hook_if_declared_p:n .. . h2384
\__hook_if_DEPRECATED_GENERIC:n .. .
    ..... h2407
\__hook_if_DEPRECATED_GENERIC:nTF .. .
    ..... h2399, h733, h785,
    h951, h984, h1271, h1302, h1767, h1883
\__hook_if_DEPRECATED_GENERIC:w .. .
    ..... h2408, h2409
\__hook_if_DEPRECATED_GENERIC_-_
    p:n .. . h2399
\__hook_if_disabled:n .. . h273
\__hook_if_disabled:nTF .. . h264,
    h294, h312, h521, h591, h1779,
    h1791, h1872, h1894, h1967, h2062, 223
\__hook_if_disabled_p:n .. . h264
\__hook_if_execute_immediately:n .. .
    ..... h2323
\__hook_if_execute_immediately:nTF .. .
    ..... h2253,
    h2258, h2266, h2323, h500, h582, h1278
\__hook_if_execute_immediately_-
    p:n .. . h2323
\__hook_if_file_hook:w .. .
    ..... h846, h849, h851
\__hook_if_file_hook:wTF .. .
    ..... h2233, h697, h846, 240
\__hook_if_file_hook_p:w .. . h846
\__hook_if_GENERIC:n .. . h2399
\__hook_if_GENERIC:nTF .. . h2399,
    h716, h772, h1018, h1777, h1891
\__hook_if_GENERIC:w .. h2400, h2401
\__hook_if_GENERIC_p:n .. . h2399
\__hook_if_GENERIC_reversed:n h2439
\__hook_if_GENERIC_reversed:nTF ..
    ..... h2439, h302, h320, h728, h780
\__hook_if_GENERIC_reversed:w .. .
    ..... h2440, h2441
\__hook_if_GENERIC_reversed_p:n .. .
    ..... h2439
\__hook_if_has_hash:n .. . i311
\__hook_if_has_hash:nTF .. .
    ..... i169, i244, i311, 309
\__hook_if_has_hash:w .. .
    ..... i311, i312, i313, i319, i321
\__hook_if_has_hash_check:w .. .
    ..... i311, i318, i323
\__hook_if_has_HASH_P:n .. . i311
\__hook_if_label_case:nnnn .. .
    ... h1364, h1364, h1512, h1593, h2001
\__hook_if_public_command:N .. i108, 306
\__hook_if_public_command:NTF .. i77, i87
\__hook_if_public_command:w .. .
    ..... i77, i111, i117
\__hook_if_replacing_args: .. h2453
\__hook_if_replacing_args:TF .. .
    ..... h2449,
    h2455, h2460, h2461, h2466, h2467,
    h2472, h502, h544, h555, h627, h1031
\__hook_if_reversed:n .. . h2390
\__hook_if_reversed:nTF .. .
    ..... h2390, h1441, h1473, h1812,
    h1852, h1854, h1910, h1947, h1949
\__hook_if_reversed_p:n .. . h2390
\__hook_if_STRUCTURE_EXIST:n .. h2378
\__hook_if_STRUCTURE_EXIST:nTF .. .
    ..... h2357, h2378, h2609,
    h150, h162, h931, h964, h2065, 222
\__hook_if_STRUCTURE_EXIST_P:n .. h2378
\__hook_if_usable:n .. . h2372
\__hook_if_usable:nTF .. . h2281,
    h2372, h2723, h106, h515, h529,
    h557, h585, h600, h718, h774, h805,
    h947, h980, h1430, h1462, h1789,
    h1811, h1870, h1892, h1909, h1965, 223
\__hook_if_usable_p:n .. .
    ..... h2372, h1845, h1940
\__hook_if_usable_use:n .. .
    ..... h2220, h2235, h2238, h2240, 280
\__hook_include_legacy_code_-
    chunk:n .. . h126, h142, h229,
    h229, h231, h249, h251, h1429, h1461
\__hook_init_STRUCTURE:n .. .
    ..... h119, h139,
    h146, h146, h148, h158, h160, h536,
    h607, h646, h1284, h1309, h2075, 237
\__hook_initialize_all: .. .
    ..... h2794, h1372,
    h1372, h1374, h1398, h1400, h1420
\__hook_initialize_HOOK_CODE:n .. .
    ..... h2179, h1376, h1402, h1423, h1423,
    h1425, h1455, h1457, h2130, h2151, 254

```

File Key: a=ltDIRCHK.dtx, b=ltPLAIN.dtx, c=ltVERS.dtx, d=ltLUATEX.dtx, e=LTExpl.dtx,
f=ltDEFNS.dtx, g=ltCMD.dtx, h=ltHOOKS.dtx, i=ltCMDBOOKS.dtx, j=ltALLOC.dtx, k=ltCNTRL.dtx,
l=ltERROR.dtx, m=ltPAR.dtx, n=ltPARA.dtx, o=ltMETA.dtx, p=ltSPACE.dtx, q=ltLOGOS.dtx,
r=ltFILES.dtx, s=ltOUTENC.dtx, t=ltCOUNTS.dtx, u=ltLENGTH.dtx, v=ltFSSBAS.dtx,
w=ltFSSAXES.dtx, x=ltFSSTRC.dtx, y=ltFSSCMP.dtx, z=ltFSSDCL.dtx, A=ltFSSINI.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltFNTCMD.dtx, E=ltTEXTCOMP.dtx, F=ltPAGENO.dtx,
G=ltXREF.dtx, H=ltMISCEN.dtx, I=ltMATH.dtx, J=ltLISTS.dtx, K=ltBOXES.dtx, L=ltTAB.dtx,
M=ltPICTUR.dtx, N=ltTHM.dtx, O=ltSECT.dtx, P=ltFLOAT.dtx, Q=ltIDXGLO.dtx, R=ltBIBL.dtx,
S=ltMARKS.dtx, T=ltPAGE.dtx, U=ltCLASS.dtx, V=ltKEYS.dtx, W=ltFILEHOOK.dtx,
X=ltSHIPOUT.dtx, Y=ltOUTPUT.dtx, Z=ltHYPHEN.dtx, aa=ltFINAL.dtx

```

\__hook_initialize_single:NNn . . .
    ..... h1447, h1479, h1494,
    h1494, h1496, h1573, h1575, h1577, 260
\l__hook_label_0_tl . . . . . h1489
\__hook_label_if_exist_apply:nnnTF
    . . . . . h1651, h1653, h1655, h1658, 275
\__hook_label_ordered:nn . h1356, 258
\__hook_label_ordered:nnTF . . .
    . . . . . h1325, h1331, h1337, h1356, 257
\__hook_label_ordered_p:nn . . h1356
\__hook_label_pair:nn . . .
    h1324, h1330, h1336, h1340, h1343,
    h1347, h1348, h1348, h1733, h1734, 258
\l__hook_labels_int . . .
    . . . . . h1489, h1499, h1503, h1535,
    h1556, h1580, h1584, h1616, h1637, 265
\l__hook_labels_seq . . .
    . . . . . h1489, h1498, h1504,
    h1522, h1579, h1585, h1603, h1739
\__hook_list_if_rule_exists:nnnTF
    . . . . . h1994, h2011, h2012, h2014
\__hook_list_one_rule:nnn . . .
    . . . . . h1994, h2003, h2004, h2009
\__hook_list_rules:nn . . . h1832,
    h1927, h1994, h1994, h2028, 275
\__hook_log:nN . . .
    . . . . . h1749, h1752, h1757, h1763,
    h1765, h1770, h1879, h1881, h1886
\__hook_log_cmd:n . . .
    . . . . . h1751,
    h1756, h1760, h1762, h1774, h1890
\__hook_log_line:n . . .
    . . . . . h1749, h1759, h1790,
    h1792, h1796, h1808, h1820, h1830,
    h1848, h1867, h1893, h1895, h1899,
    h1906, h1918, h1925, h1943, h1962
\__hook_log_line_indent:n . . .
    . . . . . h1749, h1761, h1798, h1804,
    h1814, h1821, h1835, h1843, h1901,
    h1904, h1912, h1919, h1930, h1938
\__hook_log_next_code:n . . .
    . . . . . h1923, h1973, h1973, h1978, h1980
\__hook_log_next_code:w h1826, h1976
\__hook_make_name:n . . . h355,
    h361, h370, h376, h376, h430, h472, 232
\__hook_make_name:w h376, h378, h382
\__hook_make_prefixes:w . . .
    . . . . . i154, i207, i282, i303, i308
\__hook_make_usable:n . . .
    . . . . . h97, h133, h318, h778, h809
\__hook_make_usable:nn . . .
    . . . . . i211, i508, h81, h102, h102,
    h104, h130, h132, h300, h723, h726, 225
\__hook_misused_if_replacing_
    args:nn . . . h2449, h2450, h2456
\__hook_msg_pair_found:nnn . . .
    . . . . . h1676, h1683, h1690,
    h1698, h1706, h1717, h1730, h1730
\g__hook_name_stack_seq . . .
    . . . . . h32, h416, h417,
    h421, h428, h442, h449, h457, h467
\__hook_new:n . . . . . h89, h91, h193
\__hook_new:nn . . . . . h67
\__hook_new:nn . . .
    . . . . . h70, h72, h73, h90, h181, h213
\__hook_new_pair:nnn h202, h204, h205
\__hook_new_reversed:n . . . h189, h191
\__hook_new_reversed:nn . . . h170,
    h173, h175, h176, h190, h196, h214
\__hook_nextl(hook) . . . . . 247
\__hook_next_gset:nn . . .
    . . . . . h2308, h154, h937,
    h1049, h1055, h1069, h2080, h2105
\c__hook_nine_parameters_t1 . . .
    . . . . . h35, h114, h1020, h1134, h1803
\__hook_normalise_code_pool:n . . .
    . . . . . h120,
    h1096, h1096, h1098, h1147, h1149, 225
\__hook_normalise_cs_args:nn . . .
    . . . . . h116, h117,
    h1072, h1072, h1074, h1092, h1094, 225
\__hook_normalise_fn:nn . . .
    . . . . . h561, h1102, h1120, h1145, 251
\__hook_normalize_hook_args:Nn . . .
    . . . . . h2254, h2259, h2267, h2753,
    h70, h72, h89, h173, h175, h189,
    h267, h291, h383, h392, h1752,
    h1757, h2044, h2050, h2057, h2101
\__hook_normalize_hook_args:Nnn . . .
    . . . . . h202, h204,
    h383, h397, h488, h494, h579, h928
\__hook_normalize_hook_args_-
    aux:Nn h383, h383, h394, h399, h407
\__hook_normalize_hook_rule_-
    args:Nnnnn . . . h383, h405, h1264
\l__hook_param_text_t1 . . .
    . . . . . i8, i175, i196, i250, i271, i300, 312
\__hook_parameter:n . . .
    . . . . . h1060, h1118, h1229,
    h1229, h1231, h1246, h1248, h1784
\c__hook_parameter_cmd . . . h917, h922
\c__hook_parameter_cmd/.after_-
    t1 . . . . . h917
\c__hook_parameter_cmd/.before_-
    t1 . . . . . h917
\__hook_parse_dot_label:n . . .
    . . . . . h334, h336, h336
\__hook_parse_dot_label:w . . .
    . . . . . h336, h346, h349

```

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=ltcntrn.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

```

\__hook_parse_dot_label_aux:w . . . . .  

    ..... h336, h352, h360  

\__hook_parse_dot_label_cleanup:w . . . . .  

    ..... h336, h356, h359  

\__hook_parse_label_default:n . . . . .  

    ..... h330, h330,  

        h395, h401, h402, h409, h410, h412  

\__hook_patch_check:Nnn . . . . .  

    ..... i77, i81, i84, i87, i97  

\__hook_patch_cmd_or_delay:Nnn . . . . .  

    ..... i29, i42, i46, i46, i56, 241  

\__hook_patch_command:Nnn . . . . .  

    ..... i56, i74, i77, i77, 305  

\__hook_patch_debug:n . . . . .  

    ..... i18, i18, i79, i80,  

        i83, i86, i89, i158, i233, i372, i407,  

        i410, i411, i416, i426, i429, i430, i435  

\__hook_patch_DeclareRobustCommand:Nnn . . . . .  

    ..... i126, i130, i130, 307  

\__hook_patch_DeclareRobustCommand_-  

    aux:Nnn . . . . .  

    ..... i132, i135  

\__hook_patch_expand_redefine:Nnnn . . . . .  

    ..... i141, i146, i151,  

        i154, i154, i156, i229, i231, i370, 307  

\__hook_patch_newcommand:Nnn . . . . .  

    ..... i127, i140, i144, i144, 307  

\l__hook_patch_num_args_int . . . . .  

    ..... i7, i159,  

        i164, i167, i181, i211, i220, i234,  

        i239, i242, i256, i505, i508, i528, i535  

\l__hook_patch_prefixes_tl . . . . .  

    ..... i8, i206, i281, i298, 312  

\__hook_patch_required_catcodes:  

    ..... i441, i441, i462, i543, i571, 320  

\__hook_patch_retokenize:Nnnn . . . . .  

    ..... i412,  

        i431, i496, i496, i498, i548, i550, 316  

\__hook_post_initialization_-  

    defs: . . . . .  

    ..... h2206, h2206,  

        h2208, h2213, h2216, h2218, h1395  

\__hook_preamble_hook:n . . . . .  

    ..... h2178, h2187, h2212, h2244, h2276,  

        h2294, h1416, h1773, h1889, h2112,  

        h2116, h2127, h2140, h2150, h2160, 280  

\__hook_print_args:n . . . . .  

    ..... h1983  

\__hook_print_args:nn .  

    ..... h1781, h1983  

\__hook_prop_gput_labeled_-  

    cleanup:nnn . . . . .  

    ..... h483, h542, h552  

\__hook_prop_gput_labeled_do:Nnn . . . . .  

    ..... h566, h569  

\__hook_prop_gput_labeled_-  

    do:Nnnn . . . . .  

    ..... h483  

\l__hook_rear_tl . . . . .  

    ..... h1489,  

        h1520, h1526, h1527, h1549, h1550,  

  

\__hook redefine_with_hooks:Nnnn . . . . .  

    ..... i154, i214, i286, i296, 310  

\l__hook_replace_text_tl . . . . .  

    ..... i8, i176, i183,  

        i184, i185, i190, i197, i224, i251,  

        i258, i259, i260, i265, i272, i292, 309  

\__hook_replacement_spec:N . . . . .  

    ..... h1155, h1161  

\__hook_replacing_args_false: . . . . .  

    ..... h2278, h2449, h2463,  

        h237, h487, h634, h1381, h2043, 238  

\__hook_replacing_args_reset: . . . . .  

    ..... h2280, h2449, h2469,  

        h243, h489, h495, h1384, h2045, h2051  

\__hook_replacing_args_true: . . . . .  

    ..... h2449, h2457, h493, h1382, h2049  

\g__hook_replacing_stack_seq h2449  

\__hook_retokenize_patch:Nnn . . . . .  

    ..... i92, i367, i367  

\l__hook_return_tl .  

    ..... h2471, h2472,  

        h25, h449, h450, h457, h461, h554,  

        h562, h567, h571, h572, h613, h616,  

        h943, h976, h1536, h1537, h1617, h1618  

\__hook_rollback_tidying: . . . . .  

    ..... h2834  

\__hook_rule_<_gset:nnn . . . . .  

    ..... h1322  

\__hook_rule_>_gset:nnn . . . . .  

    ..... h1322  

\__hook_rule_after_gset:nnn . . . . .  

    ..... h1322, h1328, h1333  

\__hook_rule_before_gset:nnn . . . . .  

    ..... h1322, h1322, h1327, 263  

\__hook_rule_gclear:nnn . . . . .  

    ..... h1285, h1310, h1345, h1346, 258  

\__hook_rule_incompatible_error_-  

    gset:nnn . . . . .  

    ..... h1339  

\__hook_rule_incompatible_warning_-  

    gset:nnm . . . . .  

    ..... h1339  

\__hook_rule_unrelated_gset:nnn . . . . .  

    ..... h1345, h1345, 258  

\__hook_rule_voids_gset:nnn . . . . .  

    ..... h1334, h1334  

\__hook_seq_cname:n . . . . .  

    ..... h1487, h1488, h1506, h1540,  

        h1587, h1621, h1679, h1686, h1744  

\__hook_set_default_hook_label:n . . . . .  

    ..... h2678, h465, h465  

\__hook_set_default_label:n . . . . .  

    ..... h465, h472, h474  

\__hook_set_normalise_fn:nn . . . . .  

    ..... h559, h1096, h1100, h1105, 251  

\__hook_str_compare:nn . . . . .  

    ..... h23, h23, h1350, h1358, h1367  

\__hook_strip_double_slash:n . . . . .  

    ..... h869, h875, h876

```

File Key: a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltruatex.dtx, e=ltxpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfnctcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

```

\__hook_strip_double_slash:w . . .
    ..... h869, h877, h878, h882
\__hook_tl_csname:n . . .
    ..... h1487, h1487, h1493,
    h1505, h1521, h1524, h1526, h1530,
    h1542, h1544, h1547, h1549, h1554,
    h1586, h1602, h1605, h1607, h1611,
    h1623, h1625, h1628, h1630, h1635,
    h1677, h1678, h1684, h1685, h1743
\__hook_tl_gclear:N . . .
    h2317, h2318, h2319, h64, h64, h66, h244,
    h259, h969, h970, h974, h1531, h1612
\__hook_tl_gput:Nn . . .
    h1442, h1444, h1474, h1476, h1537,
    h1563, h1618, h1644, h1649, h1649, 265
\__hook_tl_gput_left:Nn . . .
    ..... h58, h58, h1442, h1474
\__hook_tl_gput_right:Nn . . .
    ..... h55, h55, h57, h608, h1444,
    h1476, h1565, h1645, h1647, h2097
\__hook_tl_gset:Nn . . .
    ..... h2304, h2845, h49, h49, h51,
    h53, h54, h56, h60, h1324, h1330,
    h1336, h1340, h1343, h1466, h2096
\__hook_tl_gset_eq:NN . h63, h63, h65
\__hook_tl_set:Nn . . .
    .. h47, h47, h1111, h1505, h1586, 220
\__hook_tmp:w . . .
    h2843, h2859, h2860, i173, i179, i180, i182, i248,
    i254, i255, i257, i453, i456, i460,
    i517, i520, i541, i553, i556, i569,
    h34, h34, h418, h422, h424, h1107,
    h1118, h1133, h1139, h1142, h1802,
    h1805, h1996, h2017, h2026, h2037, 319
\l__hook_tmfa_bool . . .
    ..... h24, h1831, h1834, h1842,
    h1851, h1926, h1929, h1937, h1946, 272
\l__hook_tmfa_tl . . .
    ..... i166, i168, i170, i241, i243,
    i245, i373, i388, i391, i459, i462,
    i509, i513, i529, i536, i540, i543,
    i568, i571, h25, h428, h1119, h1140
\l__hook_tmfb_tl . . .
    i378, i389, i392, h25, h1108, h1115, h1142
\__hook_toplevel<hook> . . .
    247
\__hook_toplevel_gset:nn . h2309,
    h153, h936, h941, h1049, h1053, h1068
\__hook_try_declaring_generic_-
    hook:nnn . . .
        h523, h593, h649,
        h649, h651, h666, h668, h683, h685, 239
\__hook_try_declaring_generic_-
    hook:nNnn . . .
        h687, h692, h695, h695, 240
\__hook_try_declaring_generic_-
    hook:wn . . .
        h711, h713,
        h767, h769, h795, h797, h823, h825
\__hook_try_declaring_generic_-
    hook:wnTF . . .
        h653,
        h660, h670, h677, h706, h711, h762, 241
\__hook_try_declaring_generic_-
    hook_split:nNnn . . .
        ..... h695, h699, h702, h704
\__hook_try_declaring_generic_-
    next_hook:nn . . .
        .. h649, h658, h675, h690, h2067, 239
\__hook_try_file_hook:n . . .
        ..... h2220, h2228, h2231, 280
\__hook_try_patch_with_catcodes:Nnnw
        ..... i385, i402,
        i402, i404, i417, i421, i423, i436, 315
\__hook_try_put_cmd_hook:n . i21,
    i21, i23, i32, i34, h722, h777, h808
\__hook_try_put_cmd_hook:w . . .
        ..... i21, i24, i25, i35, i36
\__hook_unpatchable_cases:n i596, i598
\__hook_update_hook_code:n . . .
        ..... i226, i545, h82,
        h303, h321, h518, h588, h948, h981,
        h1289, h1314, h1371, h1371, h1376,
        h1383, h1402, h1405, h2078, h2094, 236
\__hook_use:wn . . .
    h2176,
    h2220, h2220, h2223, h2225, h2163, 280
\__hook_use_end: . h2180, h2171, h2173
\__hook_use_i_delimit_by_s_-
    mark:nw . . .
        h45, h46, h1181
\__hook_use_initialized:n . . .
    ..... h2210, h2282, h2296, h1415,
    h2112, h2117, h2119, h2144, h2165, 282
\__hook_use_initialized:nnw . . .
    ..... h2183, h2188, h2190, h2211
\__hook_use_none_delimit_by_s_-
    mark:w . . .
        h2325,
        h2392, i489, h45, h45, h1276, h1282
\__hook_use_once:n . . .
    h2267, h2292
\__hook_use_once:nn h2254, h2259,
    h2272, h2272, h2274, h2290, h2299
\__hook_use_once_clear:n . . .
    ..... h2279, h2297,
    h2301, h2301, h2305, h2313, h2315, 283
\__hook_use_once_set:n . . .
    ..... h2277, h2295, h2301, h2303, 282
\__hook_use_undefined:w h2169, h2173
\g__hook_used_prop . . .
    h31, h1377,
    h1389, h1403, h1409, h1450, h1482
\l__hook_work_prop . . .
    h30,
    h562, h1101, h1103, h1136, h1446,
    h1478, h1501, h1508, h1510, h1519,

```

File Key: a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr1.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

h1536, h1560, h1582, h1589, h1591,
 h1600, h1617, h1641, h1711, h1722, 263
hook_U?? internal commands:
 __hook_U?? 255
hook_U(hook) internal commands:
 __hook_U(hook) 222
\hookleftarrow B480
\hookrightarrow B478
hook ?? internal commands:
 __hook~?? h1251
\phantom I75
\hrule b546, p351,
 p359, p385, p393, s289, s294, B340,
 B618, K163, K168, K216, K226,
 L359, L376, M590, M600, P395, b502
\hrulefill f884, f905, b546
\hsize 1123
\hskip 1134
\hskip... 1119
\hspace p438, p442, p448, 377
\hss 976
\Hwithstroke s494, s1213, 1134
\hwithstroke s510, s1214, 1134
\hyphenation s205, 1106
\hyphenchar H461,
 f842, f845, f854, f857, f860, f865, 1132
\hyphenpenalty v686, v718, b321

I

\I U1229, U1361,
 U1451, U1471, aa255, aa555, b437
\i s247, s402, s452, s453, s454, s455,
 s456, s457, s547, s587, s588, s680,
 s682, s684, s686, s778, s1140, s1295,
 s1297, s1299, s1301, s1352, s1355,
 s1358, s1361, s1431, aa259, aa559, 1125
\ialign B337, B459, B530,
 B533, B537, B540, I168, I170,
 I189, L191, M141, b534, b536, 1084
\IeC aa357, aa361, aa468
\if 1118
if commands:
 \if:w h2326, h2340, h2393,
 h2425, i343, h997, h1155, h1180, 283
\if_case:w . g1338, h1206, h1350, h1367
\if_catcode:w i352, 314
\if_charcode:w h2443, g877
\if_cs_exist:w
 h2193, h2327, h747, h1660,
 h2016, h2121, h2129, h2146, h2167
\if_false: ... g807, g810, g2260, g2264
\if_int_compare:w n94, h1358
\if_meaning:w
 i340, i344, i345, W241, g1221

\if_mode... 353
\if_mode_horizontal: n32, n57
\if... 925
\IfBlankF g2860
\IfBlankT g2860
\IfBlankTF R19, g2860, 1147
\IfBold 586
\IfBooleanF t82, t90, g2835
\IfBooleanT g2835
\IfBooleanTF . G67, G69, G125, g2835, 189
\IfClassAtLeastTF 877
\IfClassAtLeastTF U165
\IfClassLoadedTF 877
\IfClassLoadedTF U263
\IfClassLoadedWithOptionsTF 877
\IfClassLoadedWithOptionsTF U263
\ifcsname v340,
 w440, w443, w653, w656, x128,
 z153, A42, A53, A75, A86, U1114,
 U1249, X433, Y1907, f740, f757, 498
\ifdefined b570, b634, A623, e22, e71,
 e72, e73, e74, e117, e118, e119, aa319
\IfDocumentMetadataTF o5, o17
\iff B500
\IfFileAtLeastTF 877
\IfFileAtLeastTF U165, 1150
\IfFileExists 380, 878
\IfFileExists r469, r481, r617,
 r673, r698, a181, e37, e68, e114,
 U840, W149, W185, W195, aa656, 400
\iffontchar
 s866, s1060, s1165, s1167, s1169, s1216
\IfFontSeriesContextTF
 A496, A532, A534, 586
\IfFormatAtLeastTF 877
\IfFormatAtLeastTF U165, 944
\IfHookEmptyTF .. h2722, h2825, H217, 201
\IfHookExistsTF h2723, h2824, 294
\ifincsname 929
\ifinner
 I277, I285, I305, I322, P57, P126, P315
\IfMarksEqualTF S280, S380, 850
\ifmmode 1079
\IfNoValueF g2854
\IfNoValueT g2854
\IfNoValueTF .. aa634, aa641, aa648, g2854
\ifnum 961
\ifodd z1156, M320, M344,
 M378, M400, P68, P137, Y21,
 Y140, Y614, Y672, Y986, Y989,
 Y1022, Y1025, Y1136, Y1139,
 Y1298, Y1301, Y1578, Y1581,
 Y1699, Y1702, Y1822, Y2077, Y2085

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\IfPackageAtLeastTF 877
 \IfPackageAtLeastTF U165
 \IfPackageLoadedTF 877
 \IfPackageLoadedTF U263
 \IfPackageLoadedtTF U266, U274
 \IfPackageLoadedWithOptionsTF 877
 \IfPackageLoadedWithOptionsTF ... U263
 \IfPDFManagementActiveTF ... aa715, 1078
 \IfTargetDateBefore U1679
 \ifthenelse 667
 \IfValueF g2857
 \IfValueT g2857
 \IfValueTF g2857
 \ifvbox ... Y321, Y378, Y425, Y506, Y522
 \ifvoid 1120
 \ifx 1132
 \ignorespaces
 ... p50, p144, p163, p175, p186,
 p202, p215, p485, r71, r139, r194,
 s72, v292, v302, v312, v358, z301,
 z340, H223, H230, H281, H296,
 H344, H349, H355, I313, I340, I445,
 I446, J55, J217, K154, K393, K410,
 K427, L57, L62, L68, L83, L92,
 L105, L109, L116, L123, L125,
 L134, L154, L237, L301, L303,
 L305, L332, M36, M46, M66, M75,
 M109, M120, M131, M143, M148,
 M154, N30, N32, O110, P17, P24,
 P477, P495, P513, R7, R9, X330, 549
 \ignorespacesafterend H7
 \IJ ... s250, s436, s550, s1141, aa655, 1145
 \ij ... s249, s434, s549, s1142, aa655, 1145
 \Im B314
 \imath B309
 \immediate 1082
 \in B429, B461
 in commands:
 in_callback d955
 \in_callback 47
 \include 380
 \include r218,
 r266, r268, r284, r286, r345, r397, 940
 include/.../after 936
 include/.../before 936
 include/.../end 936
 include/after 936
 include/before 936
 include/end 936
 \IncludeInRelease h2183, h2201,
 h2206, h2216, h2220, h2223, h2249,
 h2262, h2272, h2290, h2301, b557,
 h2313, h2336, h2353, h2419, h2435,
 h2492, h2521, b581, h2624, h2633,
 h2639, h2646, h2651, h2658, h2662,
 h2669, h2699, h2706, h2798, b615,
 b622, i21, i32, i154, b648, i229,
 i402, i421, i467, i492, i496, b688,
 i548, b693, i575, i578, i608, b703,
 l138, l158, l192, l205, n12, n38,
 n160, o31, p5, c71, p22, p55, p66,
 p82, p87, p99, p105, p133, p153,
 p167, p179, p192, p207, p236, p253,
 p272, p308, p334, p367, p427, p433,
 p441, p447, p461, p467, r10, r84,
 r142, r220, r257, r278, r293, r354,
 r403, r464, r474, r500, c180, r532,
 c184, r554, r572, r582, c188, r598,
 r623, c192, r628, r636, r650, r661,
 r677, r710, s77, s104, s148, s169,
 s324, s332, s353, s369, d3, t24, t30,
 t46, t69, t77, t99, t126, t161, t177,
 t185, t203, u5, u11, v24, v53, v210,
 v232, v284, v295, v305, v354, v367,
 v445, v456, v464, v496, v504, v518,
 v539, v575, v662, v724, w2, d235,
 w385, w392, w404, w416, w424,
 d258, w506, w522, w600, w613,
 w623, w630, w638, d281, w710,
 w729, w735, w739, x113, x158,
 x161, x541, x550, y2, y22, z49, z78,
 z138, z180, z210, z241, z275, z307,
 z398, z411, z417, z457, z505, z532,
 z557, z790, z838, z884, z893, z1077,
 z1088, A33, A68, A101, A126,
 A228, A252, A301, A348, A377,
 A395, A435, A469, A498, A531,
 A543, A577, A590, A621, A630,
 A655, A672, A688, A705, B63, B81,
 B100, B110, B622, B634, D27, D34,
 E607, G12, G29, G46, G62, G76,
 G96, G111, G128, G138, G150,
 G161, H10, H72, H114, H119,
 H131, H158, H177, H233, H283,
 H308, H316, H329, H335, H347,
 H352, H362, H381, H398, H422,
 H444, H471, H496, H509, H524,
 H540, H563, H573, I79, I87, I109,
 I118, I137, I144, I152, I157, I165,
 I176, I216, I233, I272, I280, I290,
 I317, I399, I408, I441, I449, I462,
 I474, I486, I495, J125, J133, K4,
 K14, K50, K70, K91, K102, K117,
 K125, K181, K189, K235, K244,
 K281, K303, K380, K398, K414,
 e2, K434, K440, K460, K468, e11,
 L60, L65, e20, L137, L157, L222,
 L227, M10, M16, M26, e58, M39,

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=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

M55, M69, M80, M89, M100, M112, M146, M151, M160, M172, M235, M250, M311, M369, M455, M464, M473, M501, M531, e112, M558, M572, M585, M596, M608, M625, M645, M652, M686, M746, e133, M804, M820, e145, O5, O20, O161, O168, O174, O186, O193, O202, e156, O225, P35, e166, P105, P206, P232, e175, P280, P294, P335, P352, e189, P406, P412, P420, P427, P434, P440, P446, P463, f5, P482, P499, P549, f13, P559, R14, R25, R61, R77, S372, T20, T48, T69, f56, U18, U23, U36, f62, U51, U63, U87, U95, U101, U126, U144, U167, U175, U187, U200, U226, U245, U265, U273, U284, U300, U314, U332, U345, U361, U381, U404, U420, U449, U461, U493, U511, U531, U549, U568, U578, U588, U600, U632, U643, U657, U667, U723, U753, U780, U812, U961, U1041, U1048, U1102, U1237, U1368, f218, f225, W5, W15, W27, W89, W144, W181, W192, W205, W249, W257, W269, W300, W323, W344, W358, a21, W366, f284, W389, W409, W434, W480, W496, W507, W534, X3, f326, a293, f354, X430, X470, X484, Y24, f382, Y54, f389, Y153, Y181, f406, Y347, Y368, Y373, Y421, f422, Y595, Y655, Y798, Y816, Y877, Y898, Y934, Y958, f460, Y1070, f472, Y1221, a310, Y1390, Y1472, Y1566, f502, f513, Y1688, Y1902, Y1925, f538, Y1939, Y1967, f546, Y2198, Y2216, f558, Y2235, Y2281, f565, aa8, aa16, aa23, aa30, aa37, aa52, aa71, aa80, aa87, aa106, aa143, aa166, aa199, aa288, aa293, aa314, aa339, aa348, aa441, f664, f681, a26, f730, f736, f766, f807, f826, f837, f854, f865, f876, f896, b49, b88, b103, b119, b125, g1022, b134, g1073, g1076, g1091, g1104, g1107, b139, b148, g1225, g1228, b154, g1256, g1259, g1268, g1276, g1279, b168, g1403, b182, b186, b220, b228, b233, b244, b289, g2860, g2866, g2894, b351, b361, h67, h86, h102, h130, h146, h158, h170, h186, h199, h219, h229, h249, h264, h282, h288, h308, h326, h483, h576, h623, h640, h649, h666, h683, h711, h767, h795, h823, h846, h849, h869, h872, b454, h885, h903, h908, h911, h917, h922, h925, h960, h993, h1042, h1049, h1065, h1072, h1092, h1096, h1147, h1151, h1167, h1171, h1199, h1202, h1225, h1229, h1246, h1254, h1258, h1267, b472, h1298, h1372, h1398, h1423, h1455, h1494, h1575, h1763, h1879, h1973, h1978, h2039, h2054, h2071, h2084, h2102, h2107, h2112, h2134, h2154, 1143
\includeonly 380
\includeonly r218, r258, r260, r279, r280, 936
\indent p464, J161, L81, 355
\IndentBox n84, n164, 346
\index 839
\index O182, Q6, Q18, S162, T25, T36, T53, T61, Y624, Y683, 848
\indexentry Q15
\inf I25
\infty B316
\initcatcodetable d91
\input 380, 878
\input r633, d18, x16, y106, a177, a180, A751, A761, A771, B10, B11, B12, B13, B14, B23, B41, B42, B46, B47, B136, B137, B138, B139, B654, B655, B656, E1129, a237, e108, e130, f22, U617, aa164, aa178, aa203, aa281, aa325, aa405, aa661, a71, 1127
\input@path 6, 1
input@path commands:
 \input@path: a242
\inputencodingname aa382, aa404, aa486, 1133
\InputIfFileExists 380, 878
\InputIfFileExists o7, r616, r639, r654, r664, r674, r728, s1546, v416, A743, A753, A763, E845, E1212, U926, U992, W143, Z8, aa275, 398
\inputlineno l214, a330, 1093
\insert P465, P484, P502, Y521, Y522, Y1887, b206, b254, b279, b281, b284, b299, b332, 342
\InsertMark S271, S376, 848
insertmark S160, 848
\int B348
int commands:
 \int_add:Nn g543, g1170
 \int_compare:nNnTF h2284, h2481, i164, i239, i375, i487,

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=ltcntr.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

z287, z326, S58, S68, S74, X55,
 X111, X142, X353, X359, X367,
 g393, g538, g804, g846, g1814,
 g2116, g2118, h1524, h1546, h1556,
 h1605, h1627, h1637, h1985, h1991
 \backslash int_compare_p:nNn h2831, h2832
 \backslash int_decr:N
 g472, g518, g1813, h1535, h1616
 \backslash int_eval:n
 e160, X355, g2130, h1207,
 h1543, h1624, h1678, h1685, h1783
 \backslash int_gdecr:N W560
 \backslash int_gincr:N . W550, X87, X102, g1006
 \backslash int_gset:N z292, z331, W546
 \backslash int_if_odd:nTF S305, S355
 \backslash int_incr:N
 ... g296, g416, g818, g888, g895,
 g1137, g1399, g1823, h1503, h1584, 147
 \backslash int_new:N i7, W545, X346,
 X348, g15, g27, g36, g1793, h1490
 \backslash int_set:Nn i159, i202,
 i234, i277, i447, i448, i478, W212,
 W226, g1028, g1038, g1234, g1243,
 g1324, g1797, g2338, g2364, h1123
 \backslash int_set_eq:NN S51
 \backslash int_step_inline:nnn
 ... i167, i181, i242, i256
 \backslash int_use:N .. i220, i528, i535, z281,
 z282, z284, z287, z314, z315, z317,
 z326, W554, X104, X116, X363,
 X371, g25, g849, g1010, g1038,
 g1155, g1159, g1160, g1243, g1390, 961
 \backslash int_value:w X48, X137
 \backslash int_zero:N g287, g381,
 g713, g717, g853, g1129, h1499, h1580
 \backslash c_max_int S51, X214, X240, X265, X292
 \backslash c_zero_int i164, i239, X91
 \backslash interdisplaylinepenalty
 ... p14, I55, I207, I393
 \backslash interfootlinepenalty b427
 \backslash interfootnotelinepenalty
 ... p19, P467, P486, P504, b427
 \backslash interlinepenalty
 ... p11, v688, H433, H436, H454,
 H457, O67, O118, O209, O232,
 P467, P486, P504, Y340, Y1157,
 Y1161, Y1323, Y1327, b331, 1081
 \backslash inteval 78
 \backslash inteval e155, e170, 78
 \backslash intextsep Y1140, Y1144,
 Y1159, Y1162, Y1169, Y1302,
 Y1308, Y1325, Y1328, Y1337, Y2341
 \backslash intop B347, B348
 \backslash iota B276

iow commands:
 \backslash iow_char:N .. h2504, h2506, h2508,
 h2513, h2516, h2518, h2528, h2564,
 h2572, h2584, h2589, h2594, h2788,
 h2790, i586, n146, n153, n154,
 n155, n156, g404, g468, g499, g571,
 g640, g652, g683, g1048, g1054,
 g1798, g2646, g2736, h1708, h1719
 \backslash iow_log:n h1751
 \backslash iow_newline:
 ... W558, g1286, g1296, g1312, g1397
 \backslash iow_now:Nn X362
 \backslash iow_term:n i19,
 i28, i39, i48, i50, i66, i70, S20, S143,
 W551, W584, g1312, h528, h599,
 h1388, h1391, h1408, h1411, h1428,
 h1460, h1539, h1558, h1559, h1561,
 h1620, h1639, h1640, h1642, h1707,
 h1718, h1732, h1738, h1739, h1740,
 h1743, h1747, h1756, h2025, h2030

\backslash ishortstack M132
 \backslash itdefault w697, A30, B94
 \backslash item I280, H358, H408, H410,
 H424, H446, I469, I481, I508, J141,
 J219, L78, N36, N38, R4, R8, 1081
 \backslash itemindent .. J9, J42, J95, J187, J208, 711
 \backslash itemitem 1081
 \backslash itemize (env.) J242
 \backslash itemize J242
 \backslash itemsep J1, J176, 711
 \backslash iterate b490, a84, a85
 \backslash itshape s447, s807, w695,
 w696, A28, A29, A554, A587, A593,
 D21, E634, N36, N38, P399, 1111

J

\backslash J aa257, aa557
 \backslash j s248, s403, s548, s779, s1150, s1365, s1451
 \backslash jmath B310
 \backslash jobname 943
 \backslash Join A725
 \backslash joinrel B471, B478, B480, B482, B484,
 B486, B488, B490, B492, B496, B498
 \backslash jot I53, I204, I404, I414

K

\backslash k ... s485, s590, s595, s617, s622, s698,
 s699, s757, s758, s812, s814, s819,
 s821, s1251, s1319, s1320, s1337,
 s1338, s1360, s1361, s1362, s1415,
 s1416, s1449, s1450, E181, E204, 1126
 \backslash kanjiskip e74
 \backslash kappa B277
 \backslash ker I27

File Key: a=ltdefns.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltcmd.dtx, g=ltcmdhooks.dtx, h=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=lthypphen.dtx, aa=ltfinal.dtx

```

\kern ..... 1096
\kern... 1119
kernel internal commands:
  \_kernel_chk_if_free_cs:N .....
    ..... g2301, g2302, g2891, 177
  \_kernel_cmd_if_xparse:NTF .....
    ..... i128, g1058, g1319,
      g1321, g2294, g2396, g2416, 1143
  \_kernel_cs_parm_from_arg_-
    count:nnTF ..... h109
  \_kernel_exp_not:w h48, h50, h56, h61
  \l_kernel_expl_bool ..... i380
  \_kernel_file_name_sanitize:n W98
  \_kernel_msg.... 1143
  \_kernel_msg.... 1145
  \_kernel_quark_new_test:N .... g48
\kerneltmpDoNotUse i440, i452, i458, i463,
  i516, i523, i544, i552, i559, i572, 317
keys commands:
  \keys_define:nn ..... V37, V46,
    V133, V141, V145, V162, V197, aa577
  \keys_if_exist:nnTF .....
    ..... V35, V75, V92, V113
  \l_keys_key_str ..... V42, V150
  \keys_set:nn ..... V50, V212, aa595
  \l_keys_usage_load_prop ..... V158
  \l_keys_usage_preamble_prop .. V193
keys internal commands:
  \_keys_option_end: .. V31, V45, V53
  \_keys_options:n ... V25, V25, V154
  \_keys_options_aux:n V25, V26, V27
  \_keys_options_class:n V67, V71, V71
  \_keys_options_class:nnn .....
    ..... V71, V83, V90
  \l_keys_options_clist ..... V23,
    V32, V49, V77, V94, V115, V125, 927
  \_keys_options_end: ..... V25
  \_keys_options_expand_module:Nn
    . V26, V134, V134, V141, V145, V212
  \_keys_options_expand_module:nN
    ..... V134, V138
  \_keys_options_global:n .....
    ..... V33, V62, V62
  \_keys_options_loaded:n .....
    ..... V54, V156, V156
  \_keys_options_loaded:nn .....
    ..... V156, V165, V170
  \l_keys_options_loading_bool ...
    ..... V24, V48, V51, V172, 927
  \_keys_options_local: .....
    ..... V34, V119, V119
  \_keys_options_package:n .....
    ..... V68, V102, V102
  \_keys_options_package:nnn .....
    ..... V102, V106, V111
  \c_keys_props_root_str .....
    ..... V9, V10
  \_keys_remove_equals:n .....
    ..... V84, V107, V130, V130
  \_keys_remove_equals:w .....
    ..... V130, V131, V132
  \_keys_tmp:nn ..... V5, V11, V13
  \l_keys_tmpa_t1 ..... V158, V160
\kill .....
  ..... L154, L162

L
\l ..... s242, s424, s530, s771, s1143,
  U1226, U1358, U1448, U1470, aa655
\l ..... s251, s426, s551, s780, s1144, aa655
\label .. G94, G129, G133, O182, S161,
  T25, T36, T53, T61, Y623, Y682, 838
\labelenumi ..... 722
\labelenumiv ..... 722
\labelformat ..... 665
\labelformat ..... .
  ..... G119, G146, G151, G157, G162, G168
\labelitemi ..... 722
\labelitemii ..... 722
\labelitemiii ..... 722
\labelitemiv ..... 722
\labelsep .. J9, J210, J216, N36, N38, 711
\labelwidth . J9, J93, J209, J211, J214, 709
\Lambda ..... B300
\lambda ..... B278
\land ..... B368, B370
\langle ..... B594
\language ..... r52, r122, H429, H569,
  Y601, Z10, b35, b82, b84, b99, 1132
\lastbox ..... v706, I193, I194,
  J130, J136, J185, O99, O132, Y307, 348
\LastDeclaredEncoding ..... .
  ..... v137, v140, aa482, 1126
\LastMark ..... S274, S378, 852
\lastnamedcs ..... f758
\lastnodetype ..... v699, v700, v701, v705
\lastpenalty ..... v702, D112, D115
\lastskip ..... .
  ..... p45, p129, p141, p160, p221, p222,
  p226, p228, p229, p241, p259, p281,
  p284, p316, p319, p320, D102,
  D105, J115, J116, J150, J151,
  M123, b514, b515, b517, b519, 1082
\LaTeX ..... q3, q15, U1188,
  U1321, U1410, X397, X403, f837, b381
\LaTeXe ..... .
  ..... q13
\latexrelease ..... 954
\LaTeXReleaseInfo ..... .
  ..... c36, c37, c40, c45, c50, H43, 39

```

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr1.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

\latexreleaseversion	c1	H426, H448, J74, K295, K316,
\lbrace	s308, B598	O207, O212, O230, O235, b416, b527
\lbrack	b441	legacy commands:
\lccode	l19, l20, l21, l22, l23,	\legacy_if:nTF S303, S324, S353, S360
	l24, s140, s1065, H514, H529, H544,	\legacyoldstylenums E4, E617
	H587, aa224, aa241, aa249, aa256,	\leq B414, B416
	aa258, aa259, aa261, aa263, aa264,	\leqno I439
	aa265, aa266, aa541, aa549, aa556,	\let 29
	aa558, aa559, aa561, aa563, 1129	\LetLtxMacro 101
\lceil	B602	\lfloor B606
\lceil	1120	\lg 14
\ldotp	B501, B504, B619	\lgroupt B608
\ldots	s322, B505, 1108	\lhd A731
\le	B416, B418	\lhook B477, B478
\leaders	b546, B340,	\lim I6
	B558, B559, B561, B562, L376,	\liminf I8
	M565, M578, M590, M600, O214, O237	\limits B539, B543, I162, I353
\leadsto	A728	\limsup I7
\leavevmode	b546, b548, p408, p422, s75, s184,	\line I269, M158, M450, M809, M826
	s289, s290, s393, s425, s429, s432,	\linebreak 360
	s479, s763, s796, D123, E98, E927,	\linebreak p9, p27
	H433, H454, H467, H478, H486,	\linepenalty b320
	H565, H575, H588, I469, I481, I508,	\lineskip B458,
	J58, J103, K8, K17, K24, K156,	I200, K297, K317, L71, L198,
	K158, K174, K202, K263, K340,	M136, M315, M374, X223, X274,
	K447, K464, K471, L178, M134,	Y626, Y685, b410, b435, b500, b535
	M314, M373, O40, O210, O222,	\lineskiplimit B458,
	O233, P530, R34, Y159, Y164,	B510, I202, I206, K283, K298,
	Y186, Y191, b505, b532, b535, 1110	K305, X224, X275, Y626, Y685,
\left	B625, B627, B629, B631, B636,	b396, b436, b500, b537, b538, 1132
	B637, B638, B639, I167, I173, I195	\linespread v316
\Leftarrow	B410, B492, B498	\linethickness M130, M810, M827
\leftarrow	B437, B439, B480, B490, B496, B550	\linewidth r30, r100, r158,
\leftarrowfill	B534, B550	I298, I324, I470, I482, I509, I513,
\leftarroweqn	I428	I532, J15, J51, J52, J54, K293,
\leftharpoondown	B453, B467	K314, L36, P266, Y148, Y207, 710
\leftharpoonup	B452	\LinkTargetOff o20
\lefthyphenmin	Z11, b371	\LinkTargetOn o20
\leftline	K498	list (env.) J34
\leftmargin J9, J52, J53, J94, J146, J148, 711	\list J34, J236, J247
\leftmargini	I461, J17, 711	\listfiles 878
\leftmarginii	J17	\listfiles r743, 217
\leftmarginiii	J17	\listparindent J9, J41, J50, 711
\leftmarginiv	J17	\literal 1102
\leftmarginv	J17	\ll B434
\leftmarginvi	J17, 711	\llap J238, J249, K502
\leftmark	T77, 852	\lmoustache B563
\Leftrightarrow	B409	\ln 15
\leftrightarrow	B436	\lnot B326, B327
\leftskip	v683, H366,	\LoadClass 876
	H372, H376, H386, H390, H394,	\LoadClass U623, U653, U874,
		U1009, U1083, U1091, U1092, 1084
\LoadClassWithOptions	876	\LoadClassWithOptions U652

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=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphe.dtx, aa=ltfinal.dtx

\LoadFontDefinitionFile v443, v469, v470, B21, B27, B28, B29, B33
 \LoadPackageWithOptions 934
 \loccount d17
 \log I3
 \loggingall b557
 \loggingoutput b553, b571, b589, b605, b619, 1125
 \LogHook h2711, 202
 \long 1112
 \Longleftarrow B492
 \longleftarrow B489
 \Longleftrightarrow B498, B500
 \longleftarrowrightarrow B496
 \longmapsto B494
 \Longrightarrow B486
 \longrightarrow B487, B494
 \loop . . . d150, d159, v697, L382, U1136, U1197, U1267, U1330, U1381, U1419, aa365, aa376, aa386, aa397, aa427, aa453, aa463, b490, a84, 1106
 \looseness b337
 \lor B369, B371
 \lower q2, B458, K214, M35, M45, M196, M305, M306, M353, M354, M409, M410
 \lowercase l26, s141, s1066, s1544, v332, v415, H518, H533, H548, H588, 1118
 \lq b439
 lrbox (env.) 725
 \lrbox K144
 \ltfilehookdate W543
 \ltfilehookversion W543
 lua commands:
 \lua_now:n X31
 \luabytecode d202
 \luachunk d210
 \luadef d182, d186, 43
 \luafunction d178, d182, d186, 43
 \luatexbase d288
 \luatexluafunction a21, a26
 \luatexversion d5, s994, a14, e73, e119

M

\M b437
 \Macro 1081
 \mag b376
 \magstep b428
 \magstephalf b428
 \makeat... 1100
 \makeatletter i376, r32, r102, r160, v421, H26, H86, O151, U617, U838, U970, Y2, f833, 1097
 \makeatother i376, U617, aa717, f833, 1097

\makebox 725
 \makebox I298, I324, K3
 \makeglossary 839
 \makeglossary r205, Q20, 1105
 \makeindex 839
 \makeindex r204, Q3, 1105
 \makelabel J45, J97, J205, J218, J238, J249, 1082
 \MakeLinkTarget o20, 359
 \MakeLowercase aa576, 1112
 \MakeRobust z888, z1083, M806, M807, M808, M809, M810, M811, M812, M813, M814, M815, M816, M817, f283, f878, f879, f880, f881, f882, f883, f884, f885, f886, f887, f888, f889, f890, f891, f892, f893, 1143
 \maketitle 803
 \MakeTitlecase aa576, 1148
 \MakeUppercase G121, G123, G148, G159, aa576, 1112
 \mapsto B444
 \mapstochar B443, B444, B494
 \marginpar P308, 344
 \marginparpush Y85, Y1838
 \marginparsep Y84, Y1849, Y1851
 \marginparwidth P341, P359, Y83, Y1851
 \mark T31, T42, T56, T64, T82, 30
 mark commands:
 \mark_debug_off: S218, S224, S235, 851
 \mark_debug_on: S218, S219, S234, 851
 \mark_if_eq:nnnn S169
 \mark_if_eq:nnnnnn S176
 \mark_if_eq:nnnnnnTF S169, 850
 \mark_if_eq:nnnnTF S169, S281, 850
 \mark_insert:nn S134, S134, S273, T28, T29, T30, T39, T40, T41, 848
 \mark_new_class:n S7, S7, S16, S271, 848
 \mark_use_first:nn S166, S166, S275, 849
 \mark_use_last:nn S166, S167, S277, 849
 \mark_use_top:nn S166, S168, S279, 849
 mark internal commands:
 __mark_class_status:nn S236, S237, S268
 __mark_debug:n S20, S143, S218, S218, S231, S299, S349, 863
 __mark_debug_gset: S218, S222, S227, S229
 __mark_error:n S123, S126, S127, S128, S131
 __mark_new_class:nn S7, S14, S17
 __mark_status:n S265, S265, S301, S351
 __mark_update_dblcol_structures: S314, S314, S371, 854

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr1.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=lttoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

```

\__mark_update_singlecol_-
  structures: . S283, S283, S369, 854
\__mark_update_structure:nn . .
  S46, S46, S286, S290, S317, S321, 854
\__mark_update_structure_-
  alias:nn S48, S112, S112, S293,
  S294, S295, S296, S326, S330, S331, 854
\__mark_update_structure_to_-
  err:n . .
  S123, S123, S327, 854
\markboth T21, T22, T49, T51, T70, T72, 851
\markright . .
  T22, T59, T73, 851
\marks . .
  d37, aa10, aa12
math (env.) . .
  I345
\math . .
  I345
\mathaccemt . .
  1131
\mathaccent . .
  z805, z853, z887, z897
\mathalpha . .
  z975, z1154, B169, B170,
  B171, B172, B173, B174, B175,
  B176, B177, B178, B179, B180,
  B181, B182, B183, B184, B185,
  B186, B187, B188, B189, B190,
  B191, B192, B193, B194, B195,
  B196, B197, B198, B199, B200,
  B201, B202, B203, B204, B205,
  B206, B207, B208, B209, B210,
  B211, B212, B213, B214, B215,
  B216, B217, B218, B219, B220,
  B221, B222, B223, B224, B225,
  B226, B227, B228, B229, B230,
  B297, B298, B299, B300, B301,
  B302, B303, B304, B305, B306,
  B307, B516, B517, B518, B519,
  B520, B521, B522, B523, B525, B528
\mathbf . .
  A14,
  A255, A308, A353, A381, A475, B151
\mathbin z1159, B232, B233, B235, B358,
  B359, B360, B361, B364, B365,
  B366, B367, B370, B371, B372,
  B373, B374, B375, B376, B377,
  B378, B379, B380, B381, B382,
  B383, B384, B385, B386, B387,
  B388, B389, B390, B391, B392,
  B393, B394, B395, B396, B397, I37
\mathcal . .
  B150
\mathchar . .
  z916,
  z960, B335, B336, B617, b533, 1131
\mathchardef . .
  j3,
  j4, j5, j6, s70, d227, z951, b21,
  b22, b23, b24, b107, b110, b111, 1104
\mathcharzero . .
  d227
\mathchoice . .
  I61
\mathclose . .
  z1162, B231,
  B240, B242, B245, B250, B256,
  B258, B260, B566, B593, B597,
  B601, B605, B611, I43, I46, I49, I52
\mathcode . .
  z948, B252, B253, B254
\mathdefaultsmode . .
  I436, I437
\mathdollar . .
  s307, B614, 1104
\mathellipsis . .
  s321, B619, 1104
\mathfontset . .
  1079
\mathgroup . .
  v14, x303, x309, x315, x316, x327,
  E643, E8, E14, E614, E1139, b79, 1079
\mathhexbox . .
  A652, b533, 1111
\mathindent I459, I471, I483, I511, I522, 1102
\mathinner . .
  B504, B508, B513, B619
\mathit . .
  w696, A29, B153, B156, B617
\mathnormal . .
  B149
\mathop . .
  z1158, B341, B342,
  B343, B344, B345, B346, B347,
  B349, B350, B351, B352, B353,
  B354, B356, B357, B357, B540, I3,
  I4, I5, I6, I7, I8, I9, I10, I11, I12,
  I13, I14, I15, I16, I17, I18, I19, I20,
  I21, I22, I23, I24, I25, I26, I27, I28,
  I29, I30, I31, I32, I33, I34, I162, I353
\mathopen . .
  z1161, B241, B244, B249, B255,
  B257, B259, B564, B595, B599,
  B603, B607, B609, I41, I44, I47, I50
\mathord . .
  z975, z1157, B236,
  B243, B246, B251, B263, B264,
  B265, B267, B268, B269, B270,
  B271, B272, B273, B274, B275,
  B276, B277, B278, B279, B280,
  B281, B282, B283, B284, B285,
  B286, B287, B288, B289, B290,
  B291, B292, B293, B294, B295,
  B296, B308, B309, B310, B311,
  B312, B313, B314, B315, B316,
  B317, B318, B319, B320, B321,
  B322, B323, B324, B325, B327,
  B328, B329, B330, B331, B332,
  B333, B334, B524, B526, B527,
  B549, B550, B553, B554, B555,
  B556, B568, B570, B572, B575,
  B577, B591, B613, B614, B615, B616
\mathpalette . .
  B457, B461, B464, I60, I69, I99, I129
\mathparagraph s310, t168, t180, B614, 1104
\mathpunct . .
  z1163, B234, B238, B501, B502, B503
\mathrel . .
  z1160, B237, B239,
  B247, B248, B261, B262, B338,
  B398, B399, B400, B401, B402,
  B403, B404, B405, B406, B407,
  B408, B409, B410, B411, B414,

```

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

B415, B418, B419, B420, B421,
 B422, B423, B424, B425, B426,
 B427, B428, B429, B430, B432,
 B433, B434, B435, B436, B437,
 B438, B441, B442, B443, B445,
 B446, B447, B448, B449, B450,
 B451, B452, B453, B454, B455,
 B457, B461, B464, B471, B473,
 B476, B477, B479, B482, B484,
 B579, B581, B583, B585, B587,
 B589, I42, I45, I48, I51, I162, I353
 $\mathit{\mathfrak{mathring}}$ B528, 1122
 $\mathit{\mathfrak{mathrm}}$ A5, A406, A449, A481, B148, 1080
 $\mathit{\mathfrak{mathsection}}$ s311, t167, t179, B614, 1120
 $\mathit{\mathfrak{mathsf}}$ A8, A411, A454, A484, B152, B155
 $\mathit{\mathfrak{mathsterling}}$ s319, B614, 1104
 $\mathit{\mathfrak{mathstrut}}$ I84, I93, I171, I172
 $\mathit{\mathfrak{mathsurround}}$ b400, b521, 1121
 $\mathit{\mathfrak{mathsymbol}}$ z953
 $\mathit{\mathfrak{mathtt}}$ A11, A416, A459, A487, B154
 $\mathit{\mathfrak{mathunderscore}}$ B614, 1105
 $\mathit{\mathfrak{mathversion}}$ v336, A614, A616, 1098
 $\mathit{\mathfrak{mathversion.}}$ 1079
 $\mathit{\mathfrak{matrix}}$ I169, I173, I180
 $\mathit{\mathfrak{max}}$ I22
 $\mathit{\mathfrak{maxdeadcycles}}$ Y7, b373
 $\mathit{\mathfrak{maxdepth}}$
 p288, r60, r129, r184, Y92, Y171,
 Y172, Y510, Y518, Y550, Y719,
 Y728, Y768, Y995, aa152, b393, 1092
 $\mathit{\mathfrak{maxdimen}}$ b554, b569, b570, b588,
 b604, b619, v670, v680, v715, v730,
 x384, x437, B458, M475, M503,
 M533, M611, M628, U1491, U1532,
 U1541, X351, X353, X409, Y293,
 Y1857, Y1877, Y1882, Y2203,
 Y2243, Y2244, Y2246, aa156,
 b309, b394, b395, b500, b538, 979
 $\mathit{\mathfrak{mbox}}$ 725
 $\mathit{\mathfrak{mbox}}$ q13, s293, s409, s568, s1165,
 A648, B506, K11, K20, K24, M52,
 P409, P416, P437, P444, b533, 1109
 $\mathit{\mathfrak{mddefault}}$ A18, A284, A290,
 A291, A292, A331, A332, A333,
 A342, A369, A385, A402, A443,
 A479, B92, B105, B107, B121, 572
 $\mathit{\mathfrak{mdseries}}$ A16, A17, A221, A277,
 A328, A329, A363, A364, A383,
 A384, A477, A478, A651, D20, 578
 $\mathit{\mathfrak{mdseries}}$ A420
 $\mathit{\mathfrak{mdseries/defaults}}$ A420
 $\mathit{\mathfrak{meaning}}$
 z627, z640, z741, z806, z853, z917,
 z1011, z1107, z1211, a222, a231,
 f240, f302, f340, f368, f451, a326, f804
 $\mathit{\mathfrak{medbreak}}$ f885, f906, b515
 $\mathit{\mathfrak{mediumseries}}$ 578
 $\mathit{\mathfrak{medmuskip}}$ B645, I36, I38, I224, I227, I241
 $\mathit{\mathfrak{medskip}}$ p401, b518
 $\mathit{\mathfrak{medskipamount}}$ p402, p404, b517
 $\mathit{\mathfrak{medspace}}$ I214
 $\mathit{\mathfrak{MessageBreak}}$ l3, l6, l13, l33, l46,
 l60, l73, l220, l222, l228, l235, o13,
 c101, s161, s988, s1549, s1552, v34,
 v35, v561, v595, w456, x20, x21,
 x67, x88, x327, x478, x498, x530,
 x546, x561, x574, y31, y33, z280,
 z313, z582, z591, z729, A61, A94,
 D144, E23, E79, E81, E100, E851,
 E853, E854, E855, E857, E859,
 E860, E861, E862, E863, E913,
 E915, E922, E929, E1144, H50,
 H90, e78, e81, e82, e83, e84, e85,
 e86, e99, e100, e101, e102, e103,
 U292, U306, U679, U690, U692,
 U694, U705, U869, U870, U871,
 U872, U882, U883, U885, U886,
 U887, U889, U891, U977, U978,
 U980, U981, U982, U984, U986,
 U1004, U1005, U1006, U1007,
 U1068, U1085, U1086, U1158,
 U1175, U1214, U1289, U1308,
 U1347, U1402, U1436, f204, U1545,
 U1547, U1629, U1632, U1645,
 U1647, f291, W489, X99, f330,
 X175, f358, X371, X477, X478,
 X479, X480, Y582, Y1998, Y2035,
 aa300, aa301, aa302, aa304, 1106
 $\mathit{\mathfrak{mho}}$ A724
 $\mathit{\mathfrak{mid}}$ B402
 $\mathit{\mathfrak{min}}$ I23
 $\mathit{\mathfrak{minipage}}$ (env.) 726
 $\mathit{\mathfrak{minipage}}$ K327
 $\mathit{\mathfrak{mit}}$ A776
 $\mathit{\mathfrak{mkern}}$ B335,
 B338, B340, B462, B471, B513,
 B514, B515, B545, B546, B547,
 B548, B549, B550, B551, B552,
 I36, I37, I40, I73, I74, O215, O238
 $\mathit{\mathfrak{mlist}}$ commands:
 $\mathit{\mathfrak{mlist_to_hlist}}$ d1004
 $\mathit{\mathfrak{mode}}$ commands:
 $\mathit{\mathfrak{mode_if_horizontal:TF}}$ n88, n93
 $\mathit{\mathfrak{mode_if_inner:TF}}$ n89
 $\mathit{\mathfrak{mode_if_math:TF}}$ z344
 $\mathit{\mathfrak{mode_if_vertical:TF}}$ n107, n119
 $\mathit{\mathfrak{models}}$ B484, 1125

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphe.dtx, aa=ltfinal.dtx

module commands:

- `\module_error` [d344](#)
- `\module_info` [d344](#)
- `\module_warning` [d344](#)
- `\module_error` [46](#)
- `\module_info` [46](#)
- `\module_warning` [46](#)
- `\modules` [d297](#)
- `\month` [c17](#), [a188](#), [U1192](#), [U1325](#), [U1414](#), [b385](#)
- `\moveright` [Y629](#), [Y688](#)
- `\mp` [B389](#)
- `\mscount` [L379](#)

msg commands:

- `\msg{...}` [1143](#)
- `\msg_error:nn` [n115](#), [n125](#), [h434](#), [h451](#)
- `\msg_error:nnn` [S11](#), [S101](#), [S132](#), [S156](#), [V200](#), [W75](#), [g281](#), [g682](#), [g2284](#), [g2288](#), [g2779](#), [g2782](#), [g2790](#), [h76](#), [h94](#), [h179](#), [h208](#), [h211](#), [h459](#), [h522](#), [h539](#), [h592](#), [h610](#), [h2063](#)
- `\msg_error:nnnn` [i43](#), [i101](#), [i397](#), [n20](#), [n33](#), [n46](#), [n58](#), [n101](#), [V41](#), [V174](#), [g395](#), [g403](#), [g427](#), [g431](#), [g467](#), [g498](#), [g513](#), [g595](#), [g602](#), [g611](#), [g639](#), [g651](#), [g670](#), [g704](#), [g1045](#), [g1700](#), [g2352](#), [g2356](#), [g2746](#), [g2760](#), [g2802](#), [g2816](#), [h112](#), [h504](#), [h633](#)
- `\msg_error:nnnn` [g626](#), [g1844](#), [g1851](#), [g2129](#), [h438](#), [h469](#), [h478](#)
- `\msg_error:nnnnnn` [h1280](#), [h1292](#), [h1317](#), [h1691](#)
- `\msg_expandable_error:nn` [h340](#)
- `\msg_expandable_error:nnn` [h2452](#), [e182](#), [g352](#), [g2848](#), [h368](#), [h1220](#)
- `\msg_expandable_error:nnnn` [W465](#), [g2021](#), [g2048](#)
- `\msg_info:nnnn` [g69](#), [g80](#), [g185](#), [g189](#)
- `\msg_line_context:` [h2533](#), [h2538](#), [h2585](#), [S20](#), [S144](#), [g2703](#), [g2708](#), [g2713](#), [g2718](#), [h560](#)
- `\msg_module_name:n` [V60](#), [V179](#), [V185](#), [V189](#)
- `\g_msg_module_name_prop` [h2476](#)
- `\g_msg_module_type_prop` [h2474](#), [h2475](#), [S183](#), [g54](#)
- `\msg_new:nnn` [h2531](#), [h2536](#), [h2581](#), [h2587](#), [h2592](#), [h2597](#), [h2602](#), [h2606](#), [h2613](#), [h2786](#), [e184](#), [V177](#), [W471](#), [g1046](#), [g2645](#), [g2700](#), [g2705](#), [g2710](#), [g2715](#), [g2720](#), [g2732](#)
- `\msg_new:nnnn` [h2477](#), [h2487](#), [h2494](#), [h2501](#), [h2510](#), [h2524](#), [h2541](#), [h2557](#), [h2570](#), [i580](#), [i590](#), [n137](#), [n148](#), [S184](#), [S192](#), [S200](#), [S208](#), [V56](#), [V182](#), [V205](#)

new commands:

- `new_attribute` [d410](#)
- `new_bytocode` [d444](#)
- `new_chunkname` [d457](#)
- `new_luafunction` [d473](#)
- `new_whatsit` [d432](#)
- `\new_attribute` [44](#)

File Key: a=`ltdirchk.dtx`, b=`ltplain.dtx`, c=`ltvers.dtx`, d=`ltluatex.dtx`, e=`ltexpl.dtx`, f=`ltdefns.dtx`, g=`ltcmd.dtx`, h=`lthooks.dtx`, i=`ltcmddhooks.dtx`, j=`ltalloc.dtx`, k=`ltcntrl.dtx`, l=`lterror.dtx`, m=`ltpar.dtx`, n=`ltpara.dtx`, o=`ltmeta.dtx`, p=`ltspace.dtx`, q=`ltlogos.dtx`, r=`ltfiles.dtx`, s=`lttoutenc.dtx`, t=`ltcounts.dtx`, u=`ltlength.dtx`, v=`ltfssbas.dtx`, w=`ltfssaxes.dtx`, x=`ltfsstrc.dtx`, y=`ltfsscmp.dtx`, z=`ltfssdcl.dtx`, A=`ltfssini.dtx`, B=`fontdef.dtx`, C=`preload.dtx`, D=`ltfntcmd.dtx`, E=`lttextcomp.dtx`, F=`ltpageno.dtx`, G=`lxref.dtx`, H=`ltmisen.dtx`, I=`ltmath.dtx`, J=`ltlists.dtx`, K=`ltboxes.dtx`, L=`lttab.dtx`, M=`ltpictur.dtx`, N=`ltthm.dtx`, O=`ltsect.dtx`, P=`ltfloat.dtx`, Q=`ltidxglo.dtx`, R=`ltbibl.dtx`, S=`ltmarks.dtx`, T=`ltpage.dtx`, U=`ltclass.dtx`, V=`ltkeys.dtx`, W=`ltfilehook.dtx`, X=`ltshipout.dtx`, Y=`ltoutput.dtx`, Z=`lthyphen.dtx`, aa=`ltfinal.dtx`

\new_bytecode 44
 \new_chunkname 44
 \new_luafunction 44
 \new_whatsit 44
 \newattribute 43
 \newattribute d74, d238
 \newbox j13,
 H489, I66, J27, K115, L16, L17,
 L18, L343, M6, M670, M675, Y86,
 Y122, Y123, Y124, b47, b314, b523
 \newcatcodetable 43
 \newcatcodetable
 ... d84, d93, d94, d120, d121, d242
 \newcommand 81
 \newcommand s4, w530, w535, w540,
 A36, A72, B51, B52, B53, B54,
 B56, B57, B59, B60, B92, B93, B94,
 B95, B96, B97, B120, B121, B122,
 H310, H311, H312, H313, M682,
 f77, W574, W575, W576, W577,
 W579, Y2329, Y2332, Y2335,
 Y2336, Y2339, Y2340, aa566, 93
 \NewCommandCopy f473, f475, g1023,
 g1074, g1092, g1105, g1108, g1226, 97
 \newcount j7, j8,
 p125, r7, t36, x25, z27, z142, z429,
 I55, I357, I358, J23, J24, J25, J26,
 J56, J226, J241, K376, L11, L12,
 L13, L14, L15, L335, L336, L337,
 M664, M665, M666, M667, M676,
 O36, O140, O141, P3, P267, P268,
 P269, P270, U1488, Y105, Y107,
 Y109, Y111, Y113, Y121, Y2024,
 Y2327, Y2330, Y2333, Y2337, aa3,
 aa4, aa5, aa91, b47, b356, b427, 1127
 \newcounter 449
 \newcounter t10
 \newdimen j10, j11, j12,
 p124, x398, x399, I53, J9, J10, J11,
 J12, J13, J14, J15, J16, J17, J18,
 J19, J20, J21, J22, K171, K172, L3,
 L5, L6, L7, L8, L166, L338, L339,
 L340, L341, M3, M4, M5, M7,
 M431, M432, M433, M434, M435,
 M436, M668, M669, M671, M672,
 M673, M674, P451, Y71, Y72,
 Y73, Y75, Y76, Y77, Y78, Y79,
 Y80, Y81, Y82, Y83, Y84, Y85,
 Y91, Y93, Y94, Y95, Y96, Y108,
 Y110, Y112, Y114, Y115, Y116,
 Y117, Y118, Y119, Y120, Y2025,
 Y2026, b47, b309, b311, b312, b426
 \NewDocumentCommand
 h2618, h2620, h2622, h2626, h2628,
 h2630, h2641, h2643, h2653, h2655,
 h2664, h2666, h2673, h2675, h2677,
 h2689, h2691, h2715, h2717, h2720,
 o20, o28, o29, o30, t79, t87, G66,
 G68, G124, V140, V142, V153,
 V211, aa632, aa639, aa646, g2740, 145
 \NewDocumentEnvironment
 ... g2512, g2519, g2776
 \newenvironment 82
 \newenvironment f146, U1190, U1323, U1412, 214
 \NewEnvironmentCopy f514, f516, 1150
 \NewExpandableDocumentCommand
 S274, S276, S278, S280, aa602, g2796
 \newfam d38, v16, b47, 1111
 \newfont A618
 \newgroup z47
 \newhelp b306
 \NewHook h2618, h2801,
 r73, r74, r75, r348, r351, r400, x150,
 A420, A421, A422, A423, A424,
 A425, A426, A427, A428, H33, H34,
 H35, H36, H37, U955, U956, W177, 213
 \NewHookPair 300
 \NewHookWithArguments .. h2624, G101, 193
 \newif j9, c73, r5, r6, v204, w401, w413,
 z15, A528, D82, E871, G3, I75,
 I76, I203, I359, J28, J29, J30, J31,
 J32, J33, J138, K326, K432, L19,
 L251, M157, M427, M428, M429,
 M430, M459, M460, O38, O124,
 U2, f168, Y97, Y98, Y99, Y100,
 Y101, Y102, Y103, Y104, f872, 1079
 \newinsert K377,
 P390, Y27, Y1856, b193, b242, 1130
 \newlabel G84, G106, G135, 1113
 \newlanguage aa284, b47
 \newlength 457
 \newlength u3
 \newline p93, p100, p106
 \newlinechar f20, b381, a75
 \newluabytecode 44
 \newluabytecode d197, d252
 \newluachunkname 44
 \newluachunkname d205, d254
 \newluacmd 43
 \newluacmd d181, 44
 \newluafunction 43
 \newluafunction .. d4, d173, d236, d248, 43
 \NewMarkClass
 ... S271, S375, aa25, aa26, aa27, 848
 \newmarks S22, aa6, 853
 \newmathalphabet y13, y109

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\NewMirroredHookPair .. [h2618](#), [h2803](#), [193](#)
 \NewMirroredHookPairWithArguments ..
 [h2624](#), [193](#)
 \NewModuleRelease [i4](#),
 n4, o3, [c152](#), E2, S4, g9, h4, [1143](#)
 \newmuskip [b47](#)
 \newpage [Y135](#), [Y141](#), [Y152](#)
 \newprotectedluacmd [44](#)
 \newprotectedluacmd ... [s1010](#), [d181](#), X28
 \newread [b47](#), [b307](#), [1082](#)
 \NewReversedHook
 [h2618](#), [h2802](#), r349, r350,
 r401, r402, U957, U958, W178, [199](#)
 \NewReversedHookWithArguments [h2624](#), [193](#)
 \newrobustcmd [96](#)
 \newsavebox [725](#)
 \newsavebox [K115](#)
 \newskip [j14](#),
 j15, j17, p404, p405, p406, p443,
 p456, u3, H407, I360, I460, J2, J3,
 J4, J5, J6, J7, J8, Y2341, Y2342,
 Y2343, Y2347, Y2348, Y2351,
 Y2352, Y2353, Y2357, Y2358,
 Y2359, [b47](#), b310, b313, b424, b425
 \newtheorem [N1](#), [1114](#)
 \newtie [s851](#), E188,
 E189, E208, E705, E1015, E1016, [637](#)
 \newtoks [j16](#), c36, n70, v351,
 v352, w499, x247, [b47](#), b306, [1130](#)
 \newwhatsit [44](#)
 \newwhatsit [d189](#), d250
 \newwrite [r3](#),
 r4, O154, Q4, Q21, [b47](#), b308, [1104](#)
 \newXeTeXintercharclass [aa35](#)
 \next [1107](#)
 \NextLinkTarget [o20](#)
 nfss internal commands:
 _nfss_init_mv_freeze:N
 z318, z343, z344
 \NG [s531](#), s1145, aa655, [1099](#)
 \ng [s552](#), s1146, aa655, [1099](#)
 \ni [B430](#), B431
 \noalign [B339](#), [B531](#), [B534](#), [B537](#),
 B538, B542, B543, I171, I172,
 I188, I191, I205, I404, I414, L224,
 L230, L359, L378, M148, M154, [680](#)
 \nobreak p60, p72, p116,
 p142, p148, p161, p174, p200, p352,
 p360, p386, p394, p415, p422, p454,
 r203, r215, s409, s435, s437, s568,
 s1165, H332, H339, K497, O90,
 O212, O213, O217, O235, O236,
 O240, P531, S153, T33, T44, T58,
 T66, Y338, Y1153, Y1319, aa208,
 aa210, aa214, aa215, aa216, aa220,
 f886, f907, b503, b506, [b508](#), [1119](#)
 \nobreakdashes [p407](#)
 \nobreakspace [p421](#)
 \nobreakspace\ [376](#)
 \NoCaseChange [aa576](#), [1148](#)
 \nocide [845](#)
 \nocite [842](#)
 \nocite [R59](#), [1106](#)
 \nocorr [D43](#), D58, D62, D65, [1094](#)
 \nocorrlist [D89](#), [D121](#)
 \noexpand [1095](#)
 \nofiles [380](#)
 \nofiles [r199](#), [676](#)
 \noindent v693, v719, O139, [1120](#)
 \nointerlineskip
 B339, B531, B534, B538,
 B542, I297, I323, M563, M566,
 M576, M578, Y1846, Y1854, [b498](#)
 \nolimits [B348](#), [B355](#), I3, I4, I5, I9, I10,
 I11, I12, I13, I14, I15, I16, I17, I18,
 I19, I20, I21, I26, I27, I28, I29, I31, I34
 \nolinebreak [360](#)
 \nolinebreak [p9](#), [p28](#)
 \nonfrenchspacing
 b709, r48, r118, r176, f887, f908, [b431](#)
 \nonscript [I36](#), I38
 \nonumber [I387](#), I426, I427
 \nopagebreak [360](#)
 \nopagebreak [p7](#), p26
 \noprotrusion [O222](#), O245
 \normalbaselines
 I167, I169, b396, b409, b410, [b435](#)
 \normalbaselineskip
 x188, K299, K318, [b424](#), b436
 \normalcolor [I352](#),
 I456, [K89](#), K368, O218, O241,
 P97, P166, Y218, Y496, Y633,
 Y643, Y692, Y702, Y2264, Y2297, [1053](#)
 \normalfont .. v671, v731, [A653](#), A673,
 A675, A684, A689, A691, A699,
 A706, A708, A714, D18, H468,
 I352, I456, O218, O241, P401, [1111](#)
 normalfont [A420](#)
 \normalize [1092](#)
 \normalineskip [K297](#), K317, [b424](#), b435
 \normalineskiplimit
 I206, K282, K298, K304, [b424](#), b436
 \normalmarginpar [P387](#)
 \normalsfcodes [r44](#),
 r46, r48, r114, r116, r118, r172,
 r174, r176, [r198](#), Y622, Y681, [1121](#)
 \normalshape [w682](#), w723, [501](#)

File Key: a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=lttoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\normalsize	r42, r112, r170, D142 , P23 , P176 , P372 , U5 , Y621 , Y680 , 196	\OnlyDescription	x5, C3																																																						
\not	B338, B412 , B413 , B435	\oalign	s327, s357, s394 , s480 , s486 , s488 , s499 , s515, s731 , s764 , s834 , s837 , s894 , s1253, A650 , B462 , B465 , b535 , 1126																																																						
\notexpanded	313	\openin	933																																																						
\notin	B461	\openout	1082																																																						
\noexpand	1084	\openup	I199 , I204																																																						
\nu	B280	\oplus	B388																																																						
\null	s327, s363 , s486 , s489 , s834 , s837, s1253 , G21 , G38 , G55 , H433 , H454, H565 , H575 , I112 , I121 , I169 , I198, O212 , O235 , X385 , b449 , 1109	\OptionNotUsed	U448 , U473 , U1023																																																						
\nulldelimiterspace	B642, b398	or commands:																																																							
\nullfont	H174	\or:	g1339, g1340, h1210, h1211, h1212 , h1213 , h1214 , h1215 , h1216 , h1217 , h1218 , h1352 , h1368																																																						
\number	c52, c61, d105 , t142 , v621 , v624 , x439 , z64, z93 , z113 , z128 , z164 , z195 , z225, z257 , A645 , f2 , f114 , U1099 , U1192, U1325 , U1414 , X351 , a89	\oslash	B385																																																						
\numberline	O72, O82 , O248 , P17 , 1083	\OT	s372																																																						
\numexpr	s1034, d82 , d105 , d157 , z150, Y36 , f747 , b189 , b205 , b215 , b246	\otimes	B386																																																						
\nunknown	d801	\outer	d21, d38, 1132																																																						
\nwarrow	B407	\outerparskip	J1																																																						
\nxt	1107	\output	Y258 , b333																																																						
O		\outputpenalty	Y260, Y274, Y297 , Y300 , Y301 , Y336 , Y1163, Y1164 , Y1329 , Y1332 , b333																																																						
\o	s244, s400 , s533 , s770 , s1130 , aa654	\oval	M450 , M453 , M812, M829																																																						
\oalign	s253, s405 , s554 , s781 , s1136 , aa654	\over	B469, I162 , I354																																																						
\obeycr	p482	\overbrace	B536																																																						
\obeyedline	b459, b463 , b486 , 29	\overfullrule	T98 , b392																																																						
\obeyedspace	b464, b467 , b487 , 30	\overleftarrow	B533																																																						
\obeylines	H439, H460, H556, H557 , Y587 , f888, f909, b452 , 29	\overrightarrow	B530																																																						
\obeyspaces	Y587, f889, f910, b452 , 30	\owns	B431, B432																																																						
\oddsidemargin	Y72, Y74 , Y615 , Y674	P																																																							
\odot	B384	\OE	s243, s399 , s532 , s769 , s1147 , aa612, aa617 , aa622 , aa654 , 1075	\P	s310, 1104	\oe	s252, s404 , s553 , s782 , s1148 , aa612, aa617 , aa622 , aa654 , 1075	package/. /after	934	\of	I67, I356	package/. /before	934	\offinterlineskip	b498	package/after	934	\oint	B355	package/before	934	\ointop	B354, B355	\PackageError	l84 , c77 , c135, c147, s1547 , E825, E877, E921	\oldstylenums	E4, E376 , E377 , E378, E379 , E380 , E381 , E382 , E383, E384 , E385 , E610 , E1136 , 413	\PackageInfo	l84 , E829, E846, E851, E867, E868, E928, E1213	\Omega	B307	\PackageNote	l136 , 1144	\omega	B290	\PackageNoteNoLine	l136	\ominus	B387	\PackageWarning	l84 , E827, E878, E1142, X476	\omit	I191, I192 , L369 , L372 , L379 , L383	\PackageWarningNoLine	l84 , s986, Y1997	\OmitIndent	n84, n163, 346	\pagebreak	360	\onecolumn	Y143	\pagebreak	p6, p7, p23, p25	File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=lttoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx			
\OE	s243, s399 , s532 , s769 , s1147 , aa612, aa617 , aa622 , aa654 , 1075	\P	s310, 1104																																																						
\oe	s252, s404 , s553 , s782 , s1148 , aa612, aa617 , aa622 , aa654 , 1075	package/. /after	934																																																						
\of	I67, I356	package/. /before	934																																																						
\offinterlineskip	b498	package/after	934																																																						
\oint	B355	package/before	934																																																						
\ointop	B354, B355	\PackageError	l84 , c77 , c135, c147, s1547 , E825, E877, E921																																																						
\oldstylenums	E4, E376 , E377 , E378, E379 , E380 , E381 , E382 , E383, E384 , E385 , E610 , E1136 , 413	\PackageInfo	l84 , E829, E846, E851, E867, E868, E928, E1213																																																						
\Omega	B307	\PackageNote	l136 , 1144																																																						
\omega	B290	\PackageNoteNoLine	l136																																																						
\ominus	B387	\PackageWarning	l84 , E827, E878, E1142, X476																																																						
\omit	I191, I192 , L369 , L372 , L379 , L383	\PackageWarningNoLine	l84 , s986, Y1997																																																						
\OmitIndent	n84, n163, 346	\pagebreak	360																																																						
\onecolumn	Y143	\pagebreak	p6, p7, p23, p25																																																						
File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=lttoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx																																																									

\pagetotal Y130
 \paperheight Y93
 \paperwidth Y93
 \par m3, m4,
 m5, n133, n169, a123, d156, v696,
 H116, H172, H332, H339, H431,
 H452, J63, J110, J127, J129, J135,
 J161, J164, K288, K309, K364,
 K394, K411, L195, L385, O41, O90,
 O220, O242, P15, P24, P249, P344,
 P478, P496, f9, f21, T91, T92,
 X401, Y168, Y195, Y259, Y1890,
 b11, b445, b461, b463, b480, b502,
 b511, b512, b513, b515, b517, b519, 346
 para commands:
 \para_end:
 n86, n86, n133, n134, n135, 355
 \g_para_indent_box n16,
 n42, n77, n79, n82, n84, n110, 349
 \para_omit_indent: n81, n85, 346
 \para_raw_end: . n106, n128, n131, 346
 \para_raw_indent: n106, n106, n129, 346
 \para_raw_noindent:
 n106, n118, n130, 346
 para internal commands:
 __para_handle_indent:
 n34, n59, n78, n78, n112
 \g__para_standard_everypar_tl ...
 n12, n69, n71, n111, n122, 352
 __para_tmp:w n61, n65
 para/after n6, 344
 para/before n6, 344
 para/begin n6, 344
 para/end n6, 344
 \paracntvalue 347
 \paragraphmark O143
 \parallel B401
 \parbox 725
 \parbox K234, 342
 \parfillskip
 ... v670, v685, v730, H368, H378,
 H387, H395, H427, H449, J76,
 K296, K317, O207, O230, b423, 354
 \parindent H368,
 H373, H378, H387, H391, H395,
 H427, H449, J50, K291, K312,
 O208, O231, b404, b527, b528, 341
 \parsep J1, J49, J90, 711
 \parseunicodedataI d123, d162
 \parseunicodedataII d124, d126
 \parseunicodedataIII d128, d134
 \parseunicodedataIV d130, d142
 \parseunicodedataV d146, d149
 \parshape J54, 349
 \parskip H343, H425, H427,
 H447, H449, I528, J49, J73, J88,
 J90, J117, J153, J172, J223, K291,
 K312, L79, Y1163, Y1331, b411, 711
 \partial B315
 \partopsep I526, J1, J61, 711
 \PassOptionsToClass 876
 \PassOptionsToClass U380
 \PassOptionsToPackage 876
 \PassOptionsToPackage U380
 \patterns s205, 1106
 \pausing b338
 \pdffilesize e71, e117
 \pdfgentounicode aa315, aa316,
 aa320, aa335, aa340, aa341, aa342
 \pdfhorigin X314
 \pdftexrevision aa321
 \pdftexversion aa319, aa320, aa321
 \pdfvariable X313, X318
 \pdfvorigin X319
 peek commands:
 \peek_meaning:NTF g2440, 180
 \peek_meaning_remove:NTF
 g1741, g1842, g2428, 180
 \peek_N_type:TF . g1747, g1771, g1803
 \peek_remove_spaces:n g1739
 \penalty p35, p38,
 p47, p282, p292, p317, p321, D118,
 H433, H436, H454, H457, I37, I207,
 I404, I414, J190, L56, P195, P199,
 P201, P217, P221, P223, R37,
 Y138, Y178, Y197, Y200, Y1161,
 Y1327, b507, b508, b509, b510,
 b511, b512, b516, b518, b520, 1081
 \perp B447
 \phantom I75
 \Phi B305
 \phi B287
 \Pi B302
 \pi B282
 picture (env.) M21
 \picture M21
 \pm B390
 \pmatrix I173, I181
 \pmod I39
 \PopDefaultHookLabel h2677, 199
 \poptabs l256, L142, L161
 \poptracing x148, x340
 \postdisplaypenalty
 p13, I468, I480, I506, b330
 \pounds s318, 1100
 \Pr I32
 pre commands:
 \pre_shipout_filter 960

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

```

\prec ..... B421
\preceq ..... B424
\predisplaypenalty ..... . p12, I467, I479, I505, b329, 1150
\predisplaysize ..... b401
\pretolerance ..... v672, v687, v732, b316
\prevdepth ..... p288, p289, p350, p355, p384, p389, I205, P196, P198, P218, P220, Y169, Y171, Y174, b498, b502, b503, 372
\PreviousTotalPages ..... X407, 979
prg commands:
  \prg_break:n ..... . g2838, g2840, g2842, g2844, g2845
  \prg_break_point: ..... g2846
  \prg_do_nothing: ..... h2213, i193, i268, i387, i611, S149, W132, W134, X166, X167, X172, X181, X185, X321, g343, g890, g1683, 139
  \prg_new_conditional:Npnn ..... . h2323, h2338, h2355, h2372, h2378, h2384, h2390, h2399, h2407, h2421, h2439, i311, S169, S176, h273, h851, h1153, h1356
  \prg_new_protected_conditional:Npnn ..... i107, i450, h713, h769, h797, h825
  \prg_replicate:nn ..... h2196, h2204, h2270, h2285, W553, g808, g851, g901, g956, g2136, g2143, h1127
  \prg_return_false: . h2333, h2349, h2367, h2376, h2382, h2388, h2396, h2405, h2414, h2417, h2431, h2446, i122, i324, i465, S174, S181, h279, h739, h791, h801, h819, h829, h842, h859, h863, h866, h1158, h1361, 241
  \prg_return_true: ..... . h2331, h2347, h2366, h2369, h2375, h2381, h2387, h2394, h2404, h2415, h2429, h2444, i121, i324, i464, S173, S180, h278, h730, h782, h817, h840, h862, h1156, h1359
\prime ..... B253, B317, I256
\ProcessedArgument ..... g324, g328, g335, g2069, g2070, g2087, g2089, g2111, g2120, g2126, g2134, g2151, g2163, g2190, g2256, g2258, g2873, 175
\ProcessKeyOptions ..... V30, V153, 926
\ProcessKeyPackageOptions ..... 1147
\ProcessList ..... g2878
\ProcessOption* ..... 1084
\ProcessOptions ..... s1568, x71, E844, E881, U474, U565, U1087, 1090
\ProcessOptions* ..... U474
\prod ..... B349
prop commands:
  \prop_clear:N ..... g1591, h1101
  \prop_const_from_keyval:Nn ..... . h905, h913, h914, h915
  \prop_gclear:N ..... . i58, h935, h968, h1377, h1403
  \prop_gclear_new:N ..... h2310, h2320
  \prop_get:NnN ..... h562, h1536, h1617
  \prop_get:NnNTF V158, g1597, h571, h613
  \prop_gpop:NnNTF ..... h943, h976
  \prop_gput:Nnn ..... h2474, h2475, h2476, i51, S183, g54, h572, h573, h615, h618, h1450, h1482
  \prop_gset_eq:NN ..... h1103
  \prop_if_empty:NTF ..... . h2343, h1432, h1464, h1797, h1900
  \prop_if_empty_p:N ..... h2360
  \prop_if_exist:NTF ..... h2342, h2380
  \prop_if_in:NnTF ..... h803, h811, h814, h831, h834, h837, h861
  \prop_map_break: . h1513, h1594, h2002
  \prop_map_function:NN i57, h1102, 251
  \prop_map_inline:Nn ..... . V193, h1389, h1409, h1501, h1508, h1510, h1582, h1589, h1591, h1741, h1800, h1903, h1997, h1999
  \prop_new:N ..... i17, g49, h30, h31, h152, h164, h1251, h1252
  \prop_put:Nnn ..... g1588, h37, h42, h1136, h1711, h1722
  \prop_set_eq:NN ..... h1446, h1478
  \prop_show:N ..... 276
\proto ..... B398
\protect ..... l246, l248, l250, l256, l262, l269, l277, l280, l286, r211, s26, s32, s51, s55, s209, s217, z690, z1238, A638, D143, G16, G33, G50, H200, H210, H245, H248, H263, H273, L264, O12, O72, O82, O164, O171, O177, P17, R5, f102, f232, f246, f255, f260, f263, f264, f266, f267, f272, f273, W282, f278, W305, f281, f282, f307, X83, X95, X125, X132, X152, f345, f373, Y600, Y659, f580, f600, aa358, 1101
\protected ..... o6, p57, p464, s308, s309, t194, d186, w682, w686, w689, w692, w695, w698, w701, w704, z1080, A573, H133, H331, I199, I388, I426, I445, I446, L56, L201, L208, M142, f7, X496, X497, X498, f455, b463, 1147
\Provide ..... 1103

```

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=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

```

\providecommand ..... h2810, h2813, s6, s981, E811,
E812, G98, G99, G100, f178, Y2008
\ProvideCommandCopy ..... 98
\ProvideDocumentCommand ..... g2740
\ProvideDocumentEnvironment ..... g2776
\ProvideExpandableDocumentCommand g2796
\ProvideHook ..... h2756, 1143
\ProvideMirroredHookPair ..... h2756
\ProvideReversedHook ..... h2756
provides commands:
    provides_module ..... d298
\provides_module ..... 46
\ProvidesClass ..... 876
\ProvidesClass ..... U360
\ProvidesExplPackage ..... W542
\ProvidesFile ..... a92,
B665, B667, B668, B669, U369, 1106
\ProvidesPackage ..... 876
\ProvidesPackage x13, E817, E849, U283,
U362, U364, U1692, W571, X491, 1088
\ProvideTextCommand ..... s3, s60, 1103
\ProvideTextCommandDefault ... s57, 1112
\Psi ..... B306
\psi ..... B289
\PushDefaultHookLabel ..... h2677, 199
\pushtabs ... l256, L139, L138, L158, L160
\pushtracing ..... x117, x321
\put ..... M56, M58,
M70, M72, M325, M326, M327,
M328, M336, M338, M351, M352,
M353, M354, M361, M364, M384,
M385, M386, M387, M393, M395,
M407, M408, M409, M410, M415,
M420, M729, M785, M813, M830, 976

Q
\qbezier ..... 767
\qbezier ..... M682, M814, M831
\qbeziermax ... M681, M707, M708, M768
\qqquad ..... p475
\quad ... p475, I168, I170, I190, O111, 1110
quark commands:
    \q_mark g1218, g1221, g1970, g1974,
g1986, g2378, g2386, g2389, g2394
    \q_nil .. g537, g1210, g1221, g1350,
g1497, g1518, g1522, g1528, g1537,
g1539, g1890, g1914, g1923, g1955,
g1970, g1974, g1986, g1992, g2018, 168
    \quark_if_nil:NTF ... g548, g1976, 168
    \quark_if_nil:nTF ..... g1347
    \quark_if_recursion_tail_stop:N ...
..... g734, g1331

R
\r ..... s236, s388, s431,
s470, s610, s637, s647, s673, s756,
s795, s1245, s1263, s1289, s1411,
s1412, E183, E205, b443, b444, 1099
\radical ..... z1104, z1107, z1137, 1131
\raggedbottom ..... T82
\raggedleft ..... H374, H392, H403, H410
\raggedright ... H369, H388, H402, H408
\raise ... s327, s359, s430, s433, s732,
s797, s895, s1253, A651, B465,

```

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfnctcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

B513, B515, I73, K482, K491,
M61, M73, M105, M117, M195,
M305, M452, M495, M526, M551,
M619, M636, M637, M740, M796
\raisebox 726
\raisebox s872, s1222, K459
\rangle B592
\RawIndent n129, n165, 346
\RawNoIndent n129
\RawNoindent n130, n166, 346
\RawParEnd n129, n167, b482, 346
\RawShipout X151, X438, 956
\rbrace s309, B596
\rbrack b441
\rceil B600
\Re B313
\read 400
\ ReadonlyShipoutCounter X346, X440, 983
\Ref 665
\Ref G112, G120,
G139, G147, G151, G158, G162, G169
\ref G10, G147, G158, P553, 722
\refstepcounter 449
\refstepcounter G113,
G141, G151, G153, G162, G164,
I350, I507, J202, N27, O59, P9, 1088
\registernumber 45
registernumber d389
\relax 1118
\relax_ 1097
\Relbar B476, B484, B486, B492, 1125
\relbar B473, B488, B490
\relpenalty b324
remove commands:
 remove_from_callback d903
\remove_from_callback 46
\RemoveFromHook h2675, h2812, 195
\removelastskip b514, b516, b518, b520
\renewcommand 81
\renewcommand B66, B68,
B70, B71, B73, B75, B77, B78, B84,
B86, B88, B89, B103, B104, B105,
B113, B114, I455, I475, I496, f124, 98
\RenewCommandCopy f473, f475, 98
\RenewDocumentCommand g2740
\RenewDocumentEnvironment g2526, g2776
\renewenvironment 82
\renewenvironment I504, I516, f152
\RenewEnvironmentCopy f514, f516, 1150
\RenewExpandableDocumentCommand
.. S377, S378, S379, S380, g2796, 188
\repeat d154, d164, v713, L382, U1140,
U1201, U1271, U1334, U1385, U1423, aa369, aa380, aa390, aa401,
aa431, aa457, aa467, b490, a84, a86
\requestedLaTeXdate
.. h2832, U1486, U1519, U1539, U1625
\requestedpatchdate U1549, U1626
\RequirePackage 876
\RequirePackage d24, U613,
U620, U663, U672, U1083, Y2005, 199
\RequirePackageWithOptions 876
\RequirePackageWithOptions U655
reserved@a commands:
 \reserved@a: r233, r305, r367, r414, U1113, U1248
reserved@b commands:
 \reserved@b: U232, U249
reserved@c commands:
 \reserved@c: r747
\reservedb 460
\RestoreAtCatcode 1098
\restorecr p482
\ReverseBoolean g2874
\reversemarginpar P387
\rfloor B604
\rgroup B608
\rhd A733
\rho B283
\rhook B479, B480
\right B625, B627, B629, B631, B636,
B637, B638, B639, I168, I173, I197
\Rightarrow B411, B486, B498
\rightarrow B438,
B440, B444, B478, B488, B496, B549
\rightarrowarrowfill B531, B547
\rightharpoondown B455
\rightharpoonup B454, B466
\righthyphenmin Z11, b371
\rightleftharpoons B464
\rightline K498
\rightmargin J9, J40, J51, 711
\rightmark T77, 852
\rightskip
.. v684, H366, H370, H376, H386,
H389, H394, H426, H448, J75,
K295, K316, O207, O230, b417, b528
\rlap s430,
s433, s797, I428, I456, K502, L81, 1111
\rm 1087
\rmdefault A6, A208, A398, A407, A439,
A450, A482, B50, B120, E7, E613, 577
\rmfamily A4, A5, A405,
A448, A449, A480, A481, D15, 1110
rmfamily A420
\rmmath 1080
\rmoustache B565

File Key: a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntr1.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\rmsubstdefault . B18, B30, E28, E39, E85
 \Roman 449
 \Roman t138, 961
 \roman 449
 \roman t137
 \romannumeral t143, t144,
 H199, H216, H262, J43, J234, J245, 681
 \root I66, I356
 \rootbox I66
 \rq b439
 \rule 726
 \rule K393,
 K410, K427, K433, P477, P495, P513

S

\S s311, 1104
 \samepage 360
 \samepage p11, p29, 1081
 \SaveAtCatcode 1098
 \savebox 725
 \savebox K116
 \savecatcodetable d117, d168, d170
 \sb I212
 \sbox 725
 \sbox q4, s498, s514, J205, K122,
 K129, K133, K138, K143, b522, 1087

scan commands:
 \scan_new:N h44
 \scan_stop: h2440, h2441, i452, i453,
 i516, i517, i552, i553, n87, S161,
 S162, S163, V52, V64, X44, X325,
 X327, X328, g1736, g1737, h653,
 h660, h670, h677, h706, h714, h750,
 h762, h770, h798, h826, h1155, 928

scan internal commands:
 \s_file_stop W103,
 W105, W108, W110, W112, W125
 \s_hook_mark
 h2176, h2225, h2233, h2331, h2333,
 h2394, h2396, h2400, h2401, h2408,
 h2409, h2422, h2423, i24, i26, i35,
 i37, i113, i118, i312, i323, i457, i460,
 i472, i476, i481, i521, i541, i557,
 i569, h44, h45, h46, h346, h349,
 h352, h356, h359, h360, h697, h743,
 h745, h753, h755, h758, h760, h852,
 h877, h878, h882, h1176, h1193,
 h1197, h1239, h1240, h1295, h2163, 220
 \s_keys_stop V131, V132
 \scdefault w694, A27, B94
 \scriptfont x338
 \scriptscriptfont x339
 \scriptscriptstyle I65, I68
 \scriptspace b399

\scriptstyle B337, I64
 \scshape
 s300, w692, w693, A25, A26, D23, 1121
 \searrow B406
 \sec I20
 \secdef O142
 \secondoftwo 395
 \sectionmark O143, 851
 \selectfont q7,
 s302, s329, s360, s449, s809, s871,
 s1221, s1255, s1569, v291, v301,
 v311, w528, w533, w538, w684,
 w688, w691, w694, w697, w700,
 w703, w706, x114, x115, x159,
 x162, x164, A6, A9, A12, A15, A18,
 A21, A24, A27, A30, A264, A287,
 A325, A345, A360, A371, A382,
 A385, A409, A414, A419, A452,
 A457, A462, A476, A479, A482,
 A485, A488, A564, A641, A665,
 A682, A697, E36, E94, E104, E636,
 E673, E916, E933, P403, P424, 218
 selectfont x150

seq commands:
 \seq_clear:N h1498, h1579
 \seq_clear_new:N h1506, h1587
 \seq_gpop:NN h2471
 \seq_gpop:NNTF W72, h449, h457
 \seq_gpop_right:NN h428
 \seq_gpush:Nn h2459, h2465, W64, h442
 \seq_gput_right:Nn
 S23, h108, h137, h417, h421
 \seq_if_empty:NTF h416, h467
 \seq_if_exist:NTF W60
 \seq_if_in:NNTF S9, S136
 \seq_map_inline:Nn
 h2854, S77, S102, S113,
 S124, S267, S332, g2088, h1378,
 h1404, h1522, h1540, h1603, h1621
 \seq_mapthread_function:NNN ... 179
 \seq_new:N h2449,
 S6, W61, g2072, h28, h33, h1489
 \seq_put_right:Nn
 h1504, h1585, h1679, h1686
 \seq_set_split:Nnn g2086
 \seq_use:Nnnn h1739, h1744

seq internal commands:
 \g_mark_classes_seq
 S6, S9, S23, S77,
 S102, S113, S124, S136, S267, S332, 855

\series 1080
 \seriesdefault
 s1570, z390, A220, A222, A660,
 A678, A694, A711, A773, B120, 1138

File Key: a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltruatex.dtx, e=ltxexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\setattribute 44
\setattribute d82, d239
\setbox... 1119
\setbox0 1115
\setcounter 449
\setcounter .. r445, t2, t37, z143, J225,
Y2328, Y2331, Y2334, Y2338, 543
\SetDefaultHookLabel h2677, 199
\SetKeys V211, 926
\setlength 457
\setlength . p84, p244, p444, u4, I524,
I529, I530, I531, K43, K204, K265,
K268, K342, K449, K450, K451,
K480, K481, K488, K489, K490,
L176, L384, Y2344, Y2345, Y2346,
Y2349, Y2350, Y2354, Y2355,
Y2356, Y2360, Y2361, Y2362, 1123
\SetMathAlphabet
.. v11, y140, y141, z695, B155, B156
\setminus B393
\setraregcatcode .. d96, d104, d113, d114
\SetSymbolFont
..... z530, B145, B146, B147, 1084
\settodepth 457
\settodepth u17
\settoheight 457
\settoheight u17
\settowidth 457
\settowidth u17
\sf 1087
\sfcode ... b551, p417, r45, r115, r173,
aa250, aa550, b431, b432, b433, b434
\sfdefault A9, A212,
A399, A412, A440, A455, A485, B50
\sffamily A7, A8, A410,
A453, A454, A483, A484, D16, 571
\sffamily A420
\sfsubstdefault B19, B31, E30, E87
\shape 1082
\shapedefault s1570, w684, z391,
A661, A679, A695, A712, B120, 604
\sharp B330
\shipout X4,
X52, X431, X434, Y606, Y664, 1101
shipout X153
shipout commands:
\l_shipout_box ... X23, X35, X38,
X50, X61, X126, X149, X179, X184,
X207, X208, X215, X226, X229,
X230, X234, X241, X251, X259,
X266, X276, X282, X283, X286,
X293, X295, X296, X302, X304, 965
\l_shipout_box_dp_dim X194,
X197, X199, X230, X283, X512, 957
\l_shipout_box_ht_dim X193, X197,
X199, X229, X249, X282, X511, 957
\l_shipout_box_ht_plus_dp_dim
..... X196,
X199, X215, X266, X277, X279, 957
\l_shipout_box_wd_dim
. X195, X199, X241, X293, X513, 957
\shipout_debug_off: X7, X13, X415, 961
\shipout_debug_on: X7, X8, X414, 961
\shipout_discard:
..... X343, X343, X411, 960
\g_shipout_READONLY_int
. X102, X104, X111, X116, X346,
X355, X359, X363, X367, X371, 961
\g_shipout_totalpage_int 961
\g_shipout_totalpages_int
..... X87, X348, 967
shipout internal commands:
__shipout_add_background_box:n .
..... X180, X206, X206, X338, X420
__shipout_add_background_-
picture:n . X69, X337, X337, X424
__shipout_add_firstpage_-
material:Nn
..... X173, X189, X189, X413, X418
__shipout_add_firstpage_-
specials:
. X110, X166, X178, X178, X181, 970
__shipout_add_foreground_box:n .
..... X117, X257, X257, X341, X422
__shipout_add_foreground_-
picture:n . X64, X340, X340, X426
__shipout_debug:n
. X7, X7, X20, X103, X115, X148, 964
\g__shipout_debug_bool
..... X6, X10, X15, X21
__shipout_debug_gset:
..... X7, X11, X16, X18
\g__shipout_discard_bool
..... X81, X88, X90, X203, X344
__shipout_drop_firstpage_-
specials:
. X127, X167, X178, X183, X185, 970
__shipout_excuse_extra_page: ...
..... X384, X392, X392
__shipout_execute: X46, X46, X52, 965
__shipout_execute_cont:
..... X57, X59, X59
__shipout_execute_main_cont:Nnnn
..... X60, X77, X77, X147, 966
__shipout_execute_nohooks_cont:
..... X144, X146
__shipout_execute_raw:
..... X135, X135, X151, 969

File Key: a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

__shipout_execute_test_level: ...
 X49, X54, X54
 __shipout_execute_test_level_-
 raw: X135, X138, X141
 __shipout_finalize_box:
 X26, X28, X44, X124
 \l__shipout_firstpage_box
 X170, X180, X187, 971
 __shipout_get_box_size:N
 ... X85, X107, X192, X192, X207, 972
 \l__shipout_group_level_tl
 X47, X53, X56, X136, X143
 \c__shipout_horigin_tl .. X310, X325
 __shipout_init_page_origins: ...
 X310, X310, X321, X324
 \g__shipout_lastpage_handled_-
 bool X120, X188, X365
 __shipout_picture_overlay:n ...
 X323, X323, X338, X341
 \l__shipout_raw_box X25,
 X139, X147, X149, X179, X184, 968
 __shipout_run_firstpage_hook: ...
 X108, X163, X163, X172, 970
 \l__shipout_saved_badness_tl ...
 X204, X210,
 X218, X231, X236, X244, X253,
 X261, X269, X281, X288, X298, X306
 __shipout_saved_protect: ...
 X83, X132, X152, X152
 \l__shipout_tmp_box
 X204, X217, X219, X220,
 X221, X225, X243, X245, X246,
 X247, X250, X268, X270, X271,
 X272, X278, X297, X299, X300,
 X301, X303, X329, X331, X332, X333
 \c__shipout_vorigin_tl .. X310, X327
 shipout/after X153, 957
 shipout/background X153, 957
 shipout/before X153, 957
 shipout/firstpage X153, 957
 shipout/foreground X153, 957
 shipout/lastpage X153, 957
 \ShipoutBox X23, X439, X494, 959
 \ShipoutBoxDepth X459, X511
 \ShipoutBoxHeight X458, X511, 983
 \ShipoutBoxWidth X460, X511
 \shortstack M132, M147, M152, M815, M832
 \show f552, f625, 102
 show commands:
 \show_hook:n 270
 \showbox Y1919
 \showboxbreadth b554,
 b631, b655, b672, b697, Y1919, b387
 \showboxdepth .. b554, b630, b654, b671,
 b698, v672, v716, v733, Y1919, b388
 \ShowCommand f545, g1229,
 g1257, g1269, g1277, g1280, g1404, 102
 \ShowDocumentCommandArgSpec g2879
 \ShowDocumentEnvironmentArgSpec .. g2879
 \ShowEnvironment f680, 104
 \ShowFloat Y1900
 \ShowHook h2711, h2817, 208
 \showhyphens v661, 1132
 \showoutput b553, 1112
 \showoverfull
 ... b552, b555, b577, b612, b620, 1112
 \showtokens f660, 104
 \Sigma B303
 \sigma B284
 \sim B445, B457
 \simeq B446
 \sin I9
 \sinh I11
 \size 1080
 \skew B544
 \skip d33, K367, P391, Y318,
 Y494, b28, b53, b92, b209, b250, b295
 skip commands:
 \skip_eval:n e162
 \skip_set:Nn n25
 \skip_zero:N n51,
 X222, X223, X224, X273, X274, X275
 \c_zero_skip n26
 \skipdef d229, b45, b53, b92
 \skipeval 78
 \skipeval e155, e172, 78
 \skipzero d229
 \slash f890, f911, b507
 \sldefault w691, A24, B94
 \SLiTeX 1102
 \slloppy K300, K319, T86, T91
 \slloppypar (env.) T91
 \slloppypar T91
 \slshape s440,
 s800, w689, w690, A22, A23, D22, E627
 \small 196
 \smallbreak f891, f912, b515
 \smallint B357
 \smallskip p401, b516
 \smallskipamount p401, p404, b515
 \smash o25, B473,
 B547, B548, B551, B552, I126, 696
 \smile B450
 \sourceLaTeXdate . h2831, c164, U65, U103
 \sp I212
 \space b447, 1096

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfnctcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\spacefactor o24, o26, p130, p139, p158, p172, p184, p198, p212, p417, p430, p435, s70, s73, P531, P533, b505, b506, 1123
 \spacefactor_in_math_mode_gh/643 . 1146
 \spaceskip E6, E612, b421
 \spadesuit B334
 \span L383
 \special o22, 970
 \SplitArgument g2874
 \splitfirstmark Y2249
 \SplitList g2874, 190
 \splitmaxdepth P469, P488, P506, Y2243, b394
 \splittopskip ... P468, P487, P505, b419
 \sqcap B376
 \sqcup B377
 \sqrt I355, 1111
 \sqrtsign B529, I71, I355, 1111
 \sqsubset A729
 \sqsubseteqq B399
 \sqsupset A730
 \sqsupseteqq B400
 \SS s304, s534, s1157, aa655, 1116
 \ss s254, s406, s557, s784, s1132, aa655
 \sscdefault w536, w609, w706
 \sscshape .. w536, w608, w704, w705, D31
 \stackrel I353
 \stamp 1090
 \star B397
 \stepcounter 449
 \stepcounter t17, t27, v626, z48, G114, G141, G153, G164, I363, I423, I517, P452, P520, Y650, Y709, 1122
 \stockheight Y95
 \stockwidth Y95
 \stop H172
 \storedpar d156, d161
 str commands:
 \c_backslash_str i147, i398
 \c_hash_str i481, i485, i514, g1390
 \c_right_brace_str h1976
 \str_case:nn i600, aa567, g1049
 \str_case:nnTF i40, g2725
 \str_case_e:nnTF g2398
 \str_count:n i161, i236, h37, h43, h1208, h1784
 \str_gset:Nn h2696
 \str_head:n g2343
 \str_if_empty:NTF aa589
 \str_if_eq:nn 258
 \str_if_eq:NNTF W556
 \str_if_eq:nnTF ... h2227, h2426, h2480, h2545, i215, i287, i369, i525, i532, i561, i564, e137, W131, W550, W560, g361, g1116, g1121, g1495, g1867, g1880, g1882, g1912, g1932, g1934, g1953, g1980, g2008, g2010, g2035, g2037, g2060, g2177, g2189, g2217, g2454, h344, h436, h476, h532, h534, h603, h605, h720, h776, h807, h854, h933, h940, h966, h973, h1732, h1838, h1933, h2033, 168
 \str_if_eq_p:nn g421, g422, g423, g424, g866, g2225, g2247, h858, h1533, h1614
 \str_lowercase:n aa601
 \str_map_function:NN e141
 \str_new:N g19
 \str_replace_all:Nnn i513
 \str_set:Nn W400, W401, aa587, aa591, g180, g181, g223, g2276
 \str_tail:n .. g1867, g2340, g2366, 165
 \str_uppercase:n g2661
 str internal commands:
 __str_if_eq:nn h23, 219
 \strcmp U285, U301
 \stretch p458
 \string 1087
 \stripmeaning 1087
 \strut I191, I192, L29, f892, f913, b523, 218
 \strutbox x189, K393, K410, K427, L186, L187, P469, P477, P488, P495, P506, P513, b523
 \subparagraphmark O143
 \subsectionmark O143, 851
 \subset B426
 \subseteq B428
 \substdefault 1136
 \subsubsectionmark O143
 \succ B420
 \succeq B423
 \sum B350
 \sup I24
 \supereject 1114
 \suppressfloats Y2010
 \superset B425
 \supseteq B427
 \surd B336
 \swallow B408
 \swdefault w531, w607, w703
 \swshape ... w531, w606, w701, w702, D30
 \symbol s162, A619
 \symletters E8, E14, E614, E1139
 \symoperators B643
 sys commands:
 \sys_if_engine_luatex:TF X26

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

T

\T 123, s335, s337, s339, s341, s343, s345, s347, s349, s351, s374, U1466, U1470, U1471
\t s282, s741, s849, E157, E158, E185, E189, E191, E203, E206, E208, E688, E859, E1126, E1128, 1104
tabbing (env.) L71
\tabbing L71
\tabbingsep L130, L132, L166
\tabcolsep L259, L338
\tableofcontents 676
\tabskip I208, I209, I369, I372, I375, I377, I522, I535, I538, I540, L167, L192, b420, b534, 1084
tabular (env.) L174
\tabular L174
\tabular* L175
\tabularnewline L194, L207, 1108
\tan I15
\tanh I17
\tau B285
\tencirc C10, M125, M678
\tencircw C10, M128
\tenln C9, M124, M126, M677, M679
\tenlnw C9, M127, M129
\test 1090
\TeX q1, q12, 1135
TeX and L^AT_EX 2_E commands:
\...-h@k 902
\@without@substitution 513
\@ i375, l19, p426, q2, d20, d1022, U43, U58, U71, U80, U118, aa517, f833, f834, a68, 1128
\@...hook 226
\@?@? 1085
\@ k15, k19, k20, k21, k22, k24, k27, k28, k30, k31, r745, r765, x510, x512, x513, L238, L239, L240, L250, Y10, Y11, a334, a335
\@DeclareMathDelimiter 1096
\@defaultsups v610
\@enc@update s183, v259, v263, 1108
\@end r722, r723, H32, H106, H172, a225, f23, Z18, aa672, aa693, a72, 1090
\@endpbox L193, L236, L386
\@eqncr I381, I403, I413, I418, I541
\@fileswith@pti@ns U542, U561, U1019
\@hyph f24, f850
\@hyphenation s205
\@if@newlist Y603, Y648, Y661, Y707
\@ifdefinable s17, f132

\@input r620, r633, r673, H26, H86, f22, U1613, W172, W188, W196, aa331, a71, 1127
\@italiccorr D113, D117, f25, 1092
\@line K498
\@math@bgroup D131, D138
\@math@egroup D128
\@par m4, n133, n170, H172, H428, H433, H436, H450, H454, H457, J82, J85, K266, K288, K309, L199, O67, O118, f21, Y259, 1118
\@patterns s205
\@protect f266, f272, f281
\@sqrt 1117
\@startpbox L193, L236, L386
\@sverb H507, 689
\@text@case@aux aa582, aa613, aa618, aa623
\@underline K454, K457, K458
\@unprocessedoptions U931, U996, U1073
\@warning l215, 1088
\@Alph t140, t156, 109
\@DeclareEncodingSubset E45, E47, E48, E49, E50, E53, E60
\@DeclareMathDelimiter z967, z986, 1096
\@DeclareMathSizes v206, v207, v209
\@Ephack p191, P201, P223, P241, 1082
\@Ialph 1098
\@IncludeInRelease c71
\@IncludeInRelease c71
\@M p11, p12, p13, p14, p15, p16, p17, p18, p19, p120, v679, v686, x439, x452, I391, J194, L56, O67, O100, O118, O130, O209, O232, f37, f39, Y178, Y197, Y200, Y260, b21, b508, b509
\@MM P469, P488, P506, Y301, b21, 322
\@Mi j3, Y138
\@Mii j3, P53, P122, P194, P216, P241, P311, Y297, Y1163, Y1330
\@Miii j3, P55, P124, P313, Y300
\@Miv j3, P195, P201, P217, P223, Y274
\@Roman t138, t144, 1121
\@TeXversion l28, a329, 1101
\@abspage@last X105, X111, X351, X353, X355, X363, X367, X370, X409, X410, 977
\@acci A775, K290, K311, 1093
\@accii A775, K290, K311, 1098
\@acciii A775, K290, K311, 1098

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=ltcntr.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

\@acol L168, L178, L260, L261, L273, L274, L277, L294, L309, L317, L327
 \@acolampacol L258, L275, L277, L284, L292, L326, L329
 \@activechar@info Y579, 1095
 \@activechar@warning 1095
 \@addamp L251, L260, L261, L276, L290, L327, L328
 \@addfield L43, L53, L86, L93, L125, L140, L142
 \@addmarginpar Y333, Y1815
 \@addtobot Y979, Y1066, Y1133, Y1185, Y1294, Y1353
 \@addtocurcol ... Y330, Y1070, Y2003
 \@addtoblcol Y858, Y1566
 \@addtofilelist a104, a106, r64, r133, r188, r620, r742, A749, A752, A759, A762, A769, A772, W169, W188, W196, aa279, aa282, aa713, 1115
 \@addtonextcol .. Y857, Y1390, Y2004
 \@addtopreamble L311, L324, L330, L331, L332, L334, L346
 \@addtoreset t16, t39, t44, t89, t115, t118
 \@addtotopbot Y1016, Y1179, Y1347, Y1439, Y1528
 \@afterheading O92, O125, 1119
 \@afterindentfalse O45
 \@afterindenttrue O43, O124, O208, O231
 \@alph t139, t152, P399, 109
 \@ampacol L258, L275, L286, L329
 \@arabic t43, t111, t123, t136, t142, P397
 \@argarraycr L203, L204
 \@argdef f80
 \@argsrbox K478
 \@argtabularcr L210, L211
 \@array L181, L182
 \@arrayacol L168, L258
 \@arrayclassiv L169, L331
 \@arrayclassz L168, L275
 \@arraycr L170, L201, L203
 \@arrayparboxrestore K280, K322, L384
 \@arrayrule L309, L311, L315, L317, L319, L346
 \@arstrut L192, L237, L343
 \@arstrutbox L185, L218, L343, L385, 1096
 \@author O8, O32
 \@auxout r217, r223, r267, r285, r309, r342, r371, r394, r418, r433, G105, G134, O181, R7, R8, R39, R49, R57, R67, R84, X362
 \@backslashchar . l234, l236, B266, U1202, U1335, U1424, f231, f481, f630, f642, f646, f647, f652, f696, 1098
 \@badcrerr l277, 1096
 \@badend l247, H304, 1145
 \@badlinearg l267, M165, M177, M188, M189, M193, M242, M247, M257, M262, M275
 \@badmath l251, I275, I277, I282, I285, I294, I306, I311, I320, I333, I338, I464, I476, I492, I501
 \@badpoptabs l255, L85, L151
 \@badrequireerror U436, U1081
 \@badtab l258, L22, L87, L108, L114, L121, L148
 \@begin@tempboxa K27, K42, K203, K266, K479, K487, 1087
 \@begindocumenthook r61, r127, r130, r182, r185, R53, U1030, U1051, 226
 \@begindvi Y627, Y686, Y714, 963
 \@begindvibox X449, X450, Y86, Y715, 958
 \@beginparpenalty p15, I467, I479, I505, J23, J170, 711
 \@begintheorem N30, N35
 \@bezier M684, M683
 \@bibitem R3, R8
 \@biblabel R4, R97
 \@bitor Y15, Y885, Y905, Y941, Y964, Y1031, Y1115, Y1125, Y1273, Y1284, Y1426, Y1513, Y1631, Y1756
 \@botlist Y65, Y386, Y388, Y433, Y435, Y721, Y742, Y751, Y752, Y993, Y996, Y1031, Y1125, Y1284, Y1959, Y1987
 \@botnum P274, Y111, Y990, Y991, Y996, Y1000, Y1462, Y1467, Y1555, Y1562, Y1951, Y1979, Y2021
 \@botroom P275, Y112, Y993, Y996, Y1952, Y1980
 \@boxfpsbit Y2069, Y2071, Y2076
 \@break@loop 1096
 \@break@tfor k31, r593, r610, D98, 1096
 \@bsphack p37, p126, p341, p357, p375, p391, G102, G133, P52, P121, P310, Q6, Q18, Q23, Q35, R63, R80, Y1886
 \@caption P12, P14
 \@captype P5, P9, P12, P40, P88, P109, P157, Y2033
 \@car q14, s89, s110, f53
 \@carcube f55, f135, f633

File Key: a=ltlirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=lttoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\@cclv Y302, Y306, Y384, Y385, Y414, Y431, Y432, Y461, Y487, Y491, Y492, aa67, b16, 963
 \@cclvi d30, d58, U1200, U1333, U1422, b21, b57, b82, b93, b95, b99, b159, b173
 \@cdr r251, f53, f776, f777
 \@centercr H327, H365, H370, H375, H385, H389, H393, 1135
 \@centering I360, I361, I369, I372, I375, I534, I538
 \@cflb Y718
 \@cflt Y718
 \@changed@cmd s3, s63, s223, v131, v267, aa476, 1108
 \@changed@x s3, s211, s219, 1108
 \@changed@x@err 1116
 \@changed@x@mouth s211, s219, 1108
 \@charlb r439, r447
 \@charrb r441, r447
 \@chclass L271, L272, L335, L348, L353
 \@check@IncludeInRelease c71
 \@check@c f189, f191
 \@check@eq f195, f196, f200
 \@checkcommand 1100
 \@checkend H16, H78, H220, H227, H279, H294, H303, 1110
 \@chnum L279, L298, L335, L350, L351, L352
 \@circ M622, M640, M649, M654, M657
 \@circle M605, M606
 \@circlefnt M125, M128, M447, M491, M520, M545, M615, M631, M663, M678
 \@cite R36, R95
 \@cite@ofmt R44, R96
 \@citea R35, R37
 \@citeb R38, R39, R40, R43, R44, R66, R67, R68, R69, R83, R84, R85, R86
 \@citex R20, R21, R29, R34, 843
 \@citex@checkblank R17, R18, R30
 \@classi L271, L307
 \@classii L271, L321
 \@classiii L271, L326
 \@classiv L169, L180, L272
 \@classoptionslist U9, U483, U498, U499, U516, U728, U729, U757, U758, U784, U785, U1694, 1143
 \@classv L272, L332
 \@classz L168, L179, L271
 \@cline L367
 \@clnht M195, M196, M204, M206, M208, M218, M225, M273, M672
 \@clnwd M197, M203, M207, M209, M210, M672
 \@cls@pkg h2578, U292, U293, U306, U307, U870, U880, U928, U975, U1005, U1057, U1066, U1068, U1085, U1547, U1622, U1644, U1674
 \@clexception U31, U156, U164, U222, U322, U338, U350, U432, U454, U465, U483, U497, U515, U614, U629, U653, U727, U756, U783, U874, U921, U948, U1009, U1022, U1058, V66, V121, 902
 \@clubpenalty r7, r25, r95, r152, J128, J196, O106, O135
 \@colht r22, r92, r149, P273, P275, P278, P284, P285, P298, P299, Y116, Y233, Y244, Y253, Y254, Y389, Y401, Y436, Y449, Y478, Y509, Y539, Y545, Y549, Y559, Y564, Y649, Y708, Y781, Y819, Y863, Y888, Y907, Y947, Y969, Y1646, Y1772, Y2134, aa155
 \@column P276, Y113, Y999, Y1044, Y1113, Y1114, Y1142, Y1150, Y1271, Y1272, Y1304, Y1316, Y1424, Y1425, Y1462, Y1467, Y1511, Y1512, Y1554, Y1561, Y1947, Y1975, Y2014, Y2189
 \@colroom r23, r93, r150, Y117, Y254, Y275, Y276, Y287, Y290, Y389, Y436, Y781, Y998, Y1043, Y1109, Y1112, Y1141, Y1266, Y1270, Y1303, Y1420, Y1423, Y1506, Y1510, Y1948, Y1976, Y2144, Y2149, Y2194, aa154, 1117
 \@combinedblfloats Y754, Y2268, Y2307
 \@combinefloats Y505, Y718
 \@comdblflelt Y754
 \@comflelt Y724, Y740, Y754
 \@compatibility 1090
 \@cons t44, P193, P215, P239, P379, f52, Y239, Y892, Y911, Y927, Y951, Y953, Y973, Y975, Y1145, Y1213, Y1309, Y1382, Y1455, Y1545, Y1648, Y1671, Y1774, Y1799, Y1816, Y1817, Y2195, b196, b214
 \@contentsline@destination O188, O190, O197, 813
 \@contfield L50, L141, L153
 \@copy@ 100
 \@copy@DeclareRobustCommand f497, f567, f588, f666, f669

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=ltcntr.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphe.dtx, aa=ltfinal.dtx

```

\@copy@newcommand ..... f318,
    f498, f567, f609, f639, f666, f672, 93
\@copytexsys ..... 1090
\@ctrerr .. 1243, t155, t159, t173, t181
\@curfield ..... L16,
    L41, L47, L51, L52, L54, L130, L131
\@curline . L16, L27, L39, L44, L53,
    L54, L55, L90, L91, L103, L128, L129
\@curr@enc ..... s154, s156
\@curr@file ..... r227,
    r228, r237, r239, r263, r271, r452,
    r471, r640, r655, U855, U1147,
    U1152, U1158, U1164, U1168,
    U1179, U1188, U1215, U1278,
    U1283, U1289, U1312, U1321,
    U1347, W267, W375, W377, 900
\@curr@file@reqd ..... W267, W377, W381, 946
\@currbox P60, P91, P95, P129, P160,
    P164, P193, P214, P215, P239,
    P257, P259, P261, P319, P322,
    P327, P331, Y215, Y216, Y227,
    Y228, Y230, Y231, Y239, Y313,
    Y314, Y857, Y858, Y1106, Y1108,
    Y1116, Y1139, Y1143, Y1145,
    Y1160, Y1201, Y1213, Y1261,
    Y1264, Y1301, Y1306, Y1309,
    Y1326, Y1371, Y1382, Y1414,
    Y1430, Y1444, Y1455, Y1497,
    Y1534, Y1545, Y1585, Y1589,
    Y1600, Y1606, Y1608, Y1612,
    Y1617, Y1626, Y1635, Y1641,
    Y1648, Y1671, Y1706, Y1710,
    Y1722, Y1729, Y1731, Y1735,
    Y1741, Y1751, Y1766, Y1774,
    Y1799, Y1817, Y1826, Y2039,
    Y2040, Y2069, Y2099, Y2104,
    Y2150, Y2153, Y2165, Y2173,
    Y2190, Y2195, b275, b276, b277
\@currdir .....
    ... a111, a133, a135, a141, a143,
    a149, a151, a156, a158, a168, a181,
    a246, a259, a272, U1126, U1152,
    U1261, U1283, U1312, U1395, 9
\@current@cmd ..... s25, v271, 1108
\@currentHref ..... G99, G107
\@currentcounter .....
    ... G113, G115, G140, G142,
    G152, I365, I519, K388, P471, 1146
\@currentlabel .....
    ... G106, G116, G135, G143, G154,
    G165, G173, I364, I518, K389,
    K406, K423, P472, P490, P508, 703
\@currentlabelname ..... G98, G107
\@currenttarget ..... G131
\@currenttitle ..... G130
\@currenvir ..... l249,
    H3, H183, H238, H255, H288, H304,
    J112, K149, U1190, U1202, U1210,
    U1214, U1221, U1323, U1335,
    U1343, U1347, U1353, U1412,
    U1424, U1432, U1436, U1442, 336
\@currenvline ..... l249,
    H184, H239, H256, H289, H305, K150
\@currext ..... U30,
    U42, U57, U70, U79, U117, U318,
    U321, U322, U337, U338, U349,
    U350, U454, U465, U476, U483,
    U497, U515, U544, U624, U637,
    U639, U647, U661, U826, U827,
    U829, U832, U834, U835, U840,
    U845, U847, U852, U858, U866,
    U872, U874, U878, U883, U889,
    U898, U901, U906, U909, U912,
    U914, U915, U917, U921, U930,
    U932, U933, U938, U941, U944,
    U948, U954, U967, U972, U973,
    U978, U984, U988, U990, U991,
    U993, U995, U997, U998, U1001,
    U1007, U1009, U1022, U1035,
    U1058, U1074, U1075, V29, V66,
    V73, V78, V81, V121, V123, V126, 901
\@currextension ..... 1084
\@currlist . P193, P215, P379, Y67,
    Y313, Y390, Y393, Y437, Y440, Y1816
\@currname .. c71, c116, c124, r754,
    U29, U41, U56, U69, U78, U116,
    U290, U292, U304, U306, U318,
    U321, U337, U349, U476, U544,
    U637, U639, U647, U661, U824,
    U827, U829, U832, U834, U835,
    U840, U842, U845, U847, U852,
    U857, U866, U870, U872, U878,
    U880, U881, U883, U889, U898,
    U900, U906, U909, U912, U914,
    U915, U919, U923, U930, U932,
    U933, U938, U941, U945, U949,
    U954, U966, U990, U991, U993,
    U995, U997, U998, U1035, U1066,
    U1068, U1075, U1085, U1622,
    U1644, U1674, V29, V42, V73,
    V78, V81, V123, V126, V140, V142,
    V153, V211, h366, h372, h427, 233
\@currnamestack .....
    ... U33, U136, W547, h424, 292
\@curroption ..... 1084
\@curroptions ..... U476,
    U484, U534, U553, U1075, U1076

```

File Key: a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

\@currpath [U15](#), [U46](#),
[U139](#), [U290](#), [U292](#), [U825](#), [U840](#),
[U847](#), [U856](#), [U898](#), [U899](#), [U926](#), [886](#)
\@currpkg@reqd
..... [U320](#), [U850](#), [U852](#), [U861](#),
[U894](#), [U908](#), [U911](#), [U926](#), [U928](#), [1140](#)
\@currsize [A639](#), [1083](#)
\@currtype [Y121](#), [Y882](#), [Y883](#), [Y884](#),
[Y885](#), [Y902](#), [Y903](#), [Y904](#), [Y905](#),
[Y1031](#), [Y1115](#), [Y1125](#), [Y1273](#),
[Y1284](#), [Y1426](#), [Y1513](#), [Y1631](#),
[Y1756](#), [Y2039](#), [Y2041](#), [Y2042](#), [Y2045](#)
\@curtab [L11](#),
[L26](#), [L86](#), [L87](#), [L88](#), [L94](#), [L95](#), [L98](#),
[L102](#), [L103](#), [L107](#), [L146](#), [L147](#), [1115](#)
\@curtabmar [L11](#), [L25](#),
[L26](#), [L38](#), [L44](#), [L89](#), [L102](#), [L106](#), [L107](#)
\@dd@r [a164](#), [a165](#)
\@dashbox [M324](#), [M325](#), [M326](#),
[M327](#), [M328](#), [M331](#), [M337](#), [M339](#),
[M349](#), [M351](#), [M352](#), [M353](#), [M354](#),
[M358](#), [M362](#), [M365](#), [M382](#), [M384](#),
[M385](#), [M386](#), [M387](#), [M390](#), [M394](#),
[M396](#), [M405](#), [M407](#), [M408](#), [M409](#),
[M410](#), [M413](#), [M417](#), [M422](#), [M674](#)
\@dashcnt [M317](#), [M319](#), [M320](#),
[M321](#), [M322](#), [M323](#), [M336](#), [M338](#),
[M341](#), [M343](#), [M344](#), [M345](#), [M347](#),
[M348](#), [M361](#), [M364](#), [M376](#), [M377](#),
[M378](#), [M379](#), [M380](#), [M381](#), [M393](#),
[M395](#), [M398](#), [M399](#), [M400](#), [M401](#),
[M403](#), [M404](#), [M416](#), [M421](#), [M674](#)
\@dashdim [M316](#), [M317](#),
[M318](#), [M319](#), [M320](#), [M322](#), [M325](#),
[M327](#), [M328](#), [M329](#), [M336](#), [M338](#),
[M340](#), [M341](#), [M342](#), [M343](#), [M344](#),
[M347](#), [M351](#), [M353](#), [M354](#), [M355](#),
[M363](#), [M366](#), [M375](#), [M376](#), [M377](#),
[M378](#), [M380](#), [M384](#), [M386](#), [M387](#),
[M388](#), [M393](#), [M395](#), [M397](#), [M398](#),
[M399](#), [M400](#), [M403](#), [M407](#), [M409](#),
[M410](#), [M411](#), [M419](#), [M424](#), [M674](#)
\@date [O9](#), [O33](#)
\@dbflt [P32](#), [P264](#), [1104](#)
\@dblarg [O54](#), [O142](#), [P12](#), [f798](#)
\@dbldeferalist .. [P239](#), [Y70](#), [Y447](#),
[Y452](#), [Y454](#), [Y820](#), [Y827](#), [Y828](#),
[Y1756](#), [Y1799](#), [Y1963](#), [Y1992](#), [1129](#)
\@dblfloat [P31](#)
\@dblfloatplacement
..... [r31](#), [r101](#), [r159](#), [P280](#), [Y403](#),
[Y451](#), [Y1944](#), [Y1972](#), [Y2273](#), [Y2313](#)
\@dblblset [P26](#)
\@dblpbot [P290](#), [P304](#), [Y2357](#)

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=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

M227, M300, M303, M324, M333,
 M383, M391, M727, M783, f26, Y1855
 \@dir a163, a166, a168, a170, a171
 \@disable@packageload@do U845, W478
 \@dischypf K289, K310, f835, f869, 1094
 \@doclarpage Y298, Y373
 \@documentclass 1086
 \@documentclasshook
 U3, U733, U761, U788, 1091
 \@doendpe H221, H228, H280, H295, J123
 \@ofilelist r751, r771, H38, H88, 1108
 \@donoparitem J144, J158
 \@dot M605, M643
 \@dotsep O215, O238
 \@dottedtocline .. O200, O226, O227
 \@downline M297, M301, M306
 \@downvector M268, M306
 \@eha ... l219, l237, l239, l241, l250,
 l252, l282, r224, r268, r286, c195,
 s52, s84, v28, v58, v102, v144, v187,
 v253, v339, x106, z25, z70, z99,
 z172, z203, z233, z265, z478, z499,
 z551, z602, z647, z652, z707, z825,
 z829, z833, z868, z872, z876, z933,
 z943, z1028, z1033, z1036, z1068,
 z1071, z1144, z1147, z1150, z1217,
 z1223, D146, E24, E81, E915, E924,
 H182, H237, H254, H287, R71,
 R88, f292, f331, f359, Y1880, Y1896
 \@ehb l219,
 l244, l270, l272, l274, Y236, Y392, Y439
 \@ehc l219, l277, l280, l286, l288, c184,
 H516, H531, H546, H559, I422,
 J220, O31, f128, f155, U1674, f481, f522
 \@ehd l219, l246,
 l254, l257, l259, l265, c102, c159,
 z118, L100, L109, e88, P6, U694, U928
 \@elt r440, s1572, s1574, t20,
 t35, t53, t56, A141, A152, A154,
 A155, A180, A505, A507, A508,
 A518, A742, B17, B25, f52, Y8,
 Y11, Y15, Y27, Y30, Y31, Y32,
 Y33, Y38, Y39, Y40, Y41, Y42,
 Y43, Y44, Y45, Y47, Y51, Y57,
 Y58, Y59, Y60, Y502, Y724, Y735,
 Y740, Y750, Y762, Y764, Y792,
 Y809, Y829, Y848, Y861, Y868,
 Y919, Y922, Y931, Y1922, Y1934, 575
 \@empty k14, 882
 \@emptycol
 Y200, Y247, Y250, Y279, Y283, 1122
 \@end@check@IncludeInRelease ...
 c143, c145

\@end@tempboxa
 K36, K45, K208, K279, K485, K495
 \@enddocument@kernel@warnings
 H39, H48, H108
 \@enddocumenthook . H77, U1030, U1052
 \@endfloatbox P190, P211, P236, P248
 \@endparenv J120, J123, 719
 \@endparpenalty
 p16, I468, I480, I506, J23, J124, 711
 \@endpbox
 L193, L236, L266, L333, L384, L387
 \@endpfalse
 .. H188, H242, H259, H292, J129,
 J131, J135, J136, J138, K152, 1087
 \@endpeltrue J138
 \@endpetrue J124, J126, J134
 \@endpreamblehook 384
 \@endtheorem N13, N19, N25, N35
 \@enlargepage .. Y1865, Y1870, Y1872
 \@ensuredmath I433, I435
 \@enumctr J234, J237, J238
 \@enumdepth J226, J232, J233, J234, 722
 \@enumspacing 722
 \@eqcnt
 I357, I419, I424, I521, I536, I537, I539
 \@eqnqr ... I370, I388, I425, I426, I523
 \@eqnum I351, I352, I423, I454, I513, 1082
 \@eqnsel I357, I535, 1084
 \@eqnswfalse I387
 \@eqnswtrue I359, I366, I424, I520
 \@eqpen ... I357, I391, I393, I404, I414
 \@er@ext 1085
 \@err@ I37, I41, I44, I52, I64, I68, I71, I79
 \@esphack ... p39, p132, p346, p363,
 p380, p397, G109, G136, P385,
 Q17, Q19, Q34, R74, R90, Y1888, 1082
 \@evenfoot T12, T15, Y617, Y676
 \@evenhead T12, T15, Y616, Y675
 \@execute@begin@hook
 H185, H189, H192, 679
 \@executeoption 1085
 \@expandtwoargs c188, U224,
 U289, U482, U518, U572, U581, f217
 \@expast L239, L267
 \@expl@@filehook@clear@replacement@flag@
 U388,
 U411, W296, W319, W340, W520
 \@expl@@filehook@drop@extension@CN
 W293, W294,
 W316, W317, W337, W338, W522, 947
 \@expl@@filehook@file@pop@
 U863, W157, W526, W536
 \@expl@@filehook@file@pop@assign@nnnn
 W162, W528, 943

File Key: a=ltdefchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

\@expl@@@filehook@file@push@@
 U846, W151, W524
 \@expl@@@filehook@if@file@replaced@@TF
 W290, W313, W334, W518, 946
 \@expl@@@filehook@if@no@extension@@nTF
 W286,
 W309, W330, W508, W510, W535
 \@expl@@@filehook@normalize@file@name@@w
 W292, W315, W336, W516
 \@expl@@@filehook@resolve@file@subst@@w
 U386,
 U409, W289, W312, W333, W514
 \@expl@@@filehook@set@curr@file@@nNN
 U385,
 U408, U818, W375, W381, W512, 946
 \@expl@@@hook@curr@name@pop@@
 h2793, U89
 \@expl@@@hook@curr@name@push@@n
 1141
 \@expl@@@initialize@all@@
 h2793, H193, h1419
 \@expl@@@mark@update@dblcol@structures@@
 S368, S383, Y470
 \@expl@@@mark@update@singlocol@structures@@
 S368, S382, Y473
 \@expl@@@shipout@add@background@box@@n
 X417, X502
 \@expl@@@shipout@add@background@picture@@n
 X417, X506
 \@expl@@@shipout@add@firstpage@material@@Nn
 X417, X499
 \@expl@@@shipout@add@foreground@box@@n
 X417, X504
 \@expl@@@shipout@add@foreground@picture@@n
 X417, X508
 \@expl@char@generate@@nn
 e142, f814, 1140
 \@expl@cs@(thing)@spec@@N 1139
 \@expl@cs@argument@spec@@N
 e139, e150, f657, f710, f721, f726
 \@expl@cs@prefix@spec@@N
 e138, e149, f655, f725
 \@expl@cs@replacement@spec@@N
 e140,
 e151, f615, f650, f657, f711, f722, f726
 \@expl@cs@to@str@@N e133, e136,
 e145, e147, f489, f581, f601, f603,
 f604, f617, f630, f642, f646, f647,
 f652, f696, f705, f709, f717, f719, 1139
 \@expl@finalise@setup@@
 e30, aa268, aa269
 \@expl@pop@filename@@
 e26, U88, U92, U98, U109, 1139
 \@expl@push@filename@@
 e24, U37, U39,
 U52, U54, U64, U76, U96, U102, 880
 \@expl@push@filename@aux@@
 h2693, e25,
 U37, U48, U60, U64, U82, U102, 292
 \@expl@str@if@eq@@nnTF
 e137, e148, U289, f453, f455, 1139
 \@expl@str@map@function@@NN
 e141, e152, f811, 109
 \@expl@str@map@function@@NN and
 \@expl@char@generate@@nn 1140
 \@expl@sys@load@backend@@ r17, e21, e23
 \@expl@text@uppercase@@n 1147
 \@extra@page@added H64, X387
 \@failedlist Y846, Y869, Y885,
 Y892, Y905, Y911, Y927, Y941, Y964
 \@fcollmadefalse Y837
 \@fcollmadetrue Y925
 \@file-subst@<file> 944
 \@filef@nd r491, r503, r504, r507,
 r513, r545, r567, r590, r607, r620,
 r673, W155, W172, W188, W196, 1136
 \@filehook@file@push 900
 \@filehook@set@CurrentFile
 r315, r376, U848, W152, W364
 \@filelist
 r63, r132, r187, r741, r742, r753,
 A749, A759, A769, aa279, aa697, aa713
 \@filesfalse
 \@fileswith@pti@ns
 U436, U542, U561,
 U719, U720, U724, U726, U754,
 U755, U781, U782, U809, U1019, 1084
 \@fileswith@ptions
 U714, U715, U717, U721
 \@fileswithoptions
 U614, U621, U629, U712
 \@filestrue
 r5, U1105, U1108, U1165,
 U1169, U1240, U1243, U1298, U1302
 \@finalstrut K393, K410, K427,
 K496, L385, P477, P495, P513, 1096
 \@firststampfalse L254, L277, L294
 \@firststamptrue L262
 \@firstcolfirstmark
 Y2250, Y2251, Y2255
 \@firstcoltopmark Y2248, Y2256
 \@firstcolumnfalse Y2240, Y2285
 \@firstcolumntrue r28,
 r98, r156, Y100, Y209, Y2259, Y2291
 \@firstoffive G23, G25, G40, G57
 \@firstofone
 r126, r181, r459, s68, s153, d12,
 d100, d108, d166, x346, z53, z81,

File Key: a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

z146, z184, z214, z245, z978, H76,
 I431, e34, L372, e61, e76, P10, R38,
 R66, R83, U864, U893, f212, W353,
 W370, f477, f482, f483, f486, f487,
 f518, f523, f524, f527, f528, aa359, 98
 \@firstoftwo
 o17, c85, r506, r512, r546,
 r591, r608, s133, t191, t196, z982,
 A510, E69, E836, E887, E903, G42,
 G78, e35, e43, T16, U161, U195,
 U207, U230, U248, f212, U1683,
 f454, f456, f478, f519, f584, f635,
 f750, f760, f770, f797, f821, a90, 395
 \@firsttab
 . L2, L74, L75, L76, L106, L118, 1096
 \@flcheckspace .. Y993, Y1029, Y2140
 \@flfail
 Y869, Y920, Y941, Y951, Y964, Y973
 \@float P26, P32
 \@floatboxreset .. P101, P170, P174
 \@floatpenalty
 . P3, P53, P55, P58, P122, P124,
 P127, P191, P194, P199, P201,
 P212, P216, P221, P223, P237,
 P241, P311, P313, P317, P321, P379
 \@floatplacement
 . r31, r101, r159, P271, Y151,
 Y211, Y255, Y481, Y1945, Y1973, 1086
 \@flsetnum
 . Y990, Y1026, Y1113, Y1271,
 Y1424, Y1511, Y1582, Y1703, Y2108
 \@flsettextmin
 Y1089, Y1241, Y1410, Y1493, Y2124
 \@flstop Y2010
 \@flsucceed
 . Y862, Y870, Y919, Y953, Y975
 \@fltovf L273, P93, P162, P322
 \@flupdates .. Y996, Y1041, Y2186
 \@flushglue j17,
 H366, H370, H376, H386, H389,
 H394, H427, H449, J76, K296, K317
 \@fnssymbol t141, t160
 \@font@aliasinfo x539
 \@font@info v133, v171, v177, v391,
 v408, v646, w478, x30, x38, x46,
 x74, x87, x154, x200, x214, x225,
 x239, x255, x261, x274, x281, x288,
 x293, x303, x315, x327, x491, x503,
 x508, x515, x545, x558, x566, z279,
 z290, z299, z312, z329, z338, z353,
 z368, z426, z482, z581, z587, z631,
 z644, z727, z816, z859, z924, z1018,
 z1185, z1214, E55, E56, E99, aa478
 \@font@series@contextfalse
 A502, A538
 \@font@series@contexttrue
 A521, A525, A537
 \@font@shape@subst@warning w445,
 w448, w453, w511, w658, w661, 505
 \@font@warning v3, v560, v565, v592,
 v599, w456, x19, x33, x41, x49,
 x61, x77, x476, x490, x502, x507,
 x514, x557, x565, y30, H50, H90, aa300
 \@fontenc@load@list
 . s1573, A742, B17, B25, 1137
 \@fontswitch D126, D128, 1096
 \@footnotemark
 P454, P460, P522, P528, P529, P555
 \@footnotetext
 K358, P454, P460, P461, P538, P544
 \@for k16,
 r233, r305, r367, r414, r753, R36,
 R65, R82, U232, U249, U480, U499,
 U516, U534, U539, U553, U558,
 U594, U604, U1076, U1113, U1248, 843
 \@forced@seriesfalse w394, w407, x140
 \@forced@seriestrue w398, w409
 \@forloop k19, k20
 \@fornoop k15, k23, k29
 \@fortmp k17, k18, k26, U592,
 U594, U1112, U1113, U1247, U1248
 \@fpbot P290, P304, Y867, Y2351
 \@fpmin .. P278, P287, P301, Y115,
 Y924, Y1953, Y1981, Y2203, Y2220
 \@fps P41, P42,
 P44, P47, P64, P110, P111, P113,
 P116, P133, Y2031, Y2033, Y2036
 \@fpsadddefault
 . P45, P48, P114, P117, Y2028
 \@fpsep P289,
 P303, Y865, Y874, Y946, Y968, Y2351
 \@fpstype
 . Y987, Y1008, Y1009, Y1023,
 Y1054, Y1055, Y1079, Y1081,
 Y1084, Y1086, Y1137, Y1193,
 Y1194, Y1229, Y1232, Y1235,
 Y1238, Y1299, Y1361, Y1362,
 Y1400, Y1402, Y1405, Y1407,
 Y1481, Y1484, Y1487, Y1490,
 Y1579, Y1594, Y1596, Y1614,
 Y1623, Y1659, Y1660, Y1700,
 Y1715, Y1717, Y1737, Y1747,
 Y1786, Y1787, Y2024, Y2040,
 Y2042, Y2044, Y2047, Y2048,
 Y2049, Y2051, Y2052, Y2056,
 Y2057, Y2059, Y2060, Y2094,

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr1.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

Y2096, Y2098, Y2110, Y2112,
 Y2126, Y2128, Y2158, Y2161, Y2172
 \@fptop P288, P302, Y864, Y2351
 \@framebox@x K179, K207, K209, 1102
 \@framebox K186, K193, K197
 \@framepicbox K186, K193, K230
 \@freelist P60, P129, P319,
 P320, Y29, Y34, Y48, Y56, Y215,
 Y503, Y736, Y751, Y765, Y870,
 Y1816, Y1817, b196, b214, b275, 1130
 \@generic@error 1101
 \@generic@message 1101
 \@getcirc M437,
 M485, M514, M541, M613, M629
 \@getfpsbit
 . Y984, Y1020, Y1576, Y1697, Y2067
 \@getlarrow M266, M274, M276
 \@getlinechar M190, M229
 \@getpen p35, p38, p47, p118
 \@getrarrow M267, M274, M283
 \@glossaryfile Q21, Q22, Q31
 \@gnewline p96, p102, p109, p112
 \@gobble k6, k9, l101, l127, l147, l155,
 l180, l189, l202, o15, p76, p485,
 r64, r133, r188, c163, r466, r468,
 c187, r741, s29, d11, d98, v561,
 v594, x345, y26, z28, z30, z430,
 z441, z525, z592, z593, z622, z628,
 z636, z641, z659, z673, z683, z692,
 z705, z722, z731, z805, z807, z811,
 z819, z853, z862, z914, z916, z927,
 z1011, z1021, z1102, z1107, z1176,
 z1207, A139, A180, A518, A752,
 A762, A772, E864, H246, e33, e65,
 e88, O143, O144, O145, O146,
 O147, O182, P7, R11, R45, R46,
 f111, U682, f133, U843, U1097,
 U1161, U1188, U1292, U1321,
 U1405, U1410, U1484, f208, U1628,
 U1640, f231, f248, f252, W261, f289,
 f295, f298, f308, f328, f334, f337,
 f346, f356, f362, f365, f374, f392,
 f396, f398, f399, f401, f409, f413,
 f415, Y623, Y624, Y625, Y682,
 Y683, Y684, Y931, Y1936, Y2204,
 Y2221, aa282, aa411, aa713, 1109
 \@gobble@AddToHook@args h2809, h2810
 \@gobble@IncludeInRelease c71
 \@gobble@RemoveFromHook@arg
 . h2812, h2813
 \@gobblecr p483, p484
 \@gobblefour z24, z427,
 z583, z585, z589, z591, z601, z605,
 z729, z781, U1190, U1323, U1412, f208

\@gobblethree
 . U128, f208, f644, f656, 1108
 \@gobbletwo k12, r32,
 r102, r160, v566, v600, z132, H23,
 H51, H83, H91, e123, e128, T11,
 T13, U1160, f175, f176, U1291,
 U1404, f208, W260, W490, aa306, 1113
 \@gtempa r685,
 r686, r692, r693, r694, r719, r720,
 r722, r723, r724, L3, L5, L6, L7,
 L8, U288, U290, U303, U304, U323,
 U325, U339, U341, U351, U353,
 f126, f127, f181, f183, f575, f583, 1092
 \@halfwidth M2,
 M126, M129, M131, M227, M299,
 M302, M324, M333, M349, M361,
 M364, M383, M391, M405, M416,
 M421, M680, M706, M725, M726,
 M727, M766, M781, M782, M783
 \@halignto L170, L174, L177, L191
 \@hangfrom O66, O117, O138
 \@height p351, p359, p385,
 p393, s293, s295, x190, B340, B558,
 B559, B561, B562, K163, K168,
 K216, K226, K453, K497, L186,
 L219, L359, L376, M227, M300,
 M303, M324, M333, M351, M359,
 M383, M391, M407, M414, M590,
 M600, M726, M782, f26, Y1855, b502
 \@highpenalty p119, aa3
 \@heightab L11, L21, L23, L74,
 L86, L95, L96, L111, L146, L147, 1115
 \@hline M167, M179, M226, M265
 \@holdpg Y124,
 Y302, Y304, Y305, Y310, Y311, Y312
 \@hspace p438, p439, p455, 1083
 \@hspacer p438, p454, 1083
 \@hvector M244, M259, M265
 \@ialph 1098
 \@icentercr H344, H345
 \@iden f215
 \@if f171, f172, f174
 \@if... 1098
 \@if@DeclareRobustCommand
 . i126, f497, f554, f565,
 f566, f570, f664, f665, f668, f691, 101
 \@if@newcommand
 . i127, i138, f316, f498,
 f555, f566, f608, f620, f626, f692, 93
 \@if@newlist 1125
 \@if@pti@ns
 . U224, U227, U229, U246, U247
 \@if@ptions
 . U221, U222, U223, U878, U973

File Key: a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspacel.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\@if@short@command 1098
 \@ifatmargin L55, L106
 \@ifbothcounters t61,
 t72, t80, t88, t104, t106, t115, t117
 \@ifclasslater ... U163, U171, U179
 \@ifclassloaded .. U155, U268, U277
 \@ifclasswith U221, U270, U279
 \@ifdefinable s14,
 s17, t11, u3, A618, K115, N7, N15,
 N22, f84, f86, f130, f132, f250, 1098
 \@iffileonpath
 r487, r541, r563, r580, r601
 \@ifl@aded U155,
 U156, U157, U827, U954, U972, 900
 \@ifl@t@r
 s1538, U169, U172, U177, U180,
 U184, U188, U190, U201, U202, U703
 \@ifl@ter s1579,
 s1580, U163, U164, U183, U866, U1001
 \@ifl@ter@@ s1579, s1580
 \@ifnch e49, e50, f782, f784, f796
 \@ifnextchar
 . a101, p94, p484, r633, t13, x411,
 H343, I355, J143, K9, K11, K18,
 K20, K26, K47, K121, K122, K128,
 K129, K136, K140, K185, K186,
 K192, K193, K198, K231, K239,
 K247, K254, K258, K328, K332,
 K336, K437, K442, K465, K472,
 K477, L57, L181, L203, L210,
 e45, M23, M132, M143, M453,
 N3, N5, N28, P27, P264, P324,
 P452, P519, P536, R3, R17, R29,
 U295, U309, U698, U713, U718,
 U1106, U1109, U1241, U1244,
 Y211, Y2012, f778, f783, f797, 1126
 \@iforloop k21, k22
 \@ifpackagelater U163, U170, U178, 877
 \@ifpackageloaded
 . U155, U267, U276, Y1996, 877
 \@ifpackagewith U221, U269, U278, 877
 \@iframebox K199, K200, K201
 \@framepicbox K231, K232
 \@ifstar p60, p72,
 p331, p438, t102, t113, v206, y121,
 H332, H339, H570, H579, I390, L56,
 L202, L209, M142, M605, O52,
 O142, f73, U437, U477, Y1860, f797
 \@ifundefin@di . f740, f741, f758, f761
 \@ifundefin@di .. f740, f743, f746
 \@undefined t3, t7, t16, t50,
 t62, t64, v100, v186, x424, z472,
 E54, E866, G85, H148, H165, H181,
 H236, H253, H286, e132, N21, R40,
 R68, R85, T3, T7, U128, U153,
 U390, U396, U413, U425, U501,
 U535, U554, f127, f134, U829, f154,
 f161, f183, f194, f289, f295, W485,
 f328, f334, f356, f362, f392, f409,
 f490, f530, f531, f735, g2899, 1097
 \@ignore... 1118
 \@ignorefalse
 v362, H4, H187, H223, H230, H241,
 H258, H281, H291, H296, P384, 1082
 \@ignoretrue ... p201, p214, v358,
 H4, H7, I348, I351, I384, I544, 1082
 \@iiminipage
 . K330, K334, K337, K338, K339
 \@iiiparbox K241,
 K249, K256, K259, K260, K261, K375
 \@iiminipage K333, K335
 \@iinput r633, r634, 398
 \@iiparbox K255, K257
 \@iirsbox K477, K486
 \@imakebox K26, K41, K138
 \@imakepicbox . K47, K48, K143, K233
 \@iminipage K329, K331
 \@in@minipage@envtrue K354
 \@include .. r228, r271, r287, r291, 1138
 \@includeinreleasefalse
 . c74, c79, c133, c141, f873
 \@includeinreleasetrue c123
 \@index Q18, Q19, Q35
 \@indexfile Q4, Q5, Q14
 \@inlabel 710
 \@inlabelfalse
 . J28, J104, J184, Y165, Y192, 1093
 \@inlabeltrue J28, J178
 \@inmatherr l283, J112, J142, M605, 1100
 \@inmathwarn s3
 \@inpenc@test aa356, aa423
 \@input r34, r104, r162,
 r299, r361, r408, r672, O152, aa716, 933
 \@input@ r319, r346,
 r379, r398, r423, r674, v418, R51, 1089
 \@input@file@exists@with@hooks ..
 . W143, 1141
 \@inputcheck r482, r483,
 r490, r536, r537, r544, r558, r559,
 r566, r588, r589, r592, r605, r606,
 r609, a194, a195, a198, a206, e38,
 e39, e42, f38, f45, U1149, U1150,
 U1184, U1280, U1281, U1317,
 U1392, U1393, U1400, b307, a73, 1143
 \@insertfalse Y1077,
 Y1227, Y1398, Y1479, Y1574, Y1695
 \@inserttrue Y1003,
 Y1048, Y1165, Y1333, Y1653, Y1780

File Key: a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltruatex.dtx, e=ltxpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspac.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

\@invalidchar 1288
 \@iparbox K240, K248, K253
 \@irsbox ... K465, K472, K477, K478
 \@isavebox K136, K137
 \@isavepicbox K141, K142
 \@ishortstack M133, M141
 \@istackcr M143, M144
 \@itabcr L57, L58
 \@item J143, J156
 \@itemdepth J241, J243, J244, J245, 722
 \@itemfudge L38, L44, L82
 \@itemitem J245, J248
 \@itemlabel J44, J96, J143, 1087
 \@itempenalty p17, J23, J175, 711
 \@itemspacing 722
 \@iwhiledim k7
 \@iwhilenum k3
 \@iwhilesw k10
 \@ixpt v742
 \@ixstackcr M142
 \@kernel@... 880
 \@kernel@Ref G120, G125
 \@kernel@after@{hook} 212
 \@kernel@after@begindocument ...
 i61, r58, r76, e7, V191, 1143
 \@kernel@after@begindocument@before
 r16, r76, w731, 384
 \@kernel@after@enddocument ...
 H15, e1, X352
 \@kernel@after@enddocument@afterlastpage
 H19, e1, X358, h453
 \@kernel@after@para@after n8, n99, 345
 \@kernel@after@para@end . n8, n92, 344
 \@kernel@after@shipout@background
 X72, X160, 1142
 \@kernel@after@shipout@lastpage .
 X114, X119, X160, X378, X383, 978
 \@kernel@before@{hook} 212
 \@kernel@before@begindocument ...
 r56, r76, e7, X407, 1143
 \@kernel@before@enddocument ...
 H13, H112, 675
 \@kernel@before@insertmark ...
 S139, S160, 860
 \@kernel@before@para@before ...
 n8, n21, n47, 344
 \@kernel@before@para@begin ...
 n8, n30, n55, 344
 \@kernel@before@shipout@background
 X68, X70, X160, 966
 \@kernel@currentdata G132
 \@kernel@currpathstack ...
 U45, U47, U91, U123, 882
 \@kernel@eqno I443, I445, I451
 \@kernel@leqno I444, I446, I452
 \@kernel@make@file@csname ...
 W288, W291,
 W311, W314, W332, W335, W364
 \@kernel@pageref G65, G69
 \@kernel@pageref@exp G70
 \@kernel@ref G64, G67, G120
 \@kernel@ref@exp G72
 \@kernel@rename@newcommand ...
 . f297, f314, f351, f379, f384, f397, 93
 \@kernel@reserved@label@data ...
 G100, G107
 \@kernel@sRef G122, G125
 \@kernel@spageref ...
 G26, G43, G60, G65, G69
 \@kernel@sref G13, G25,
 G30, G42, G47, G59, G64, G67, G122
 \@killglue
 M59, M73, M103, M115, M123
 \@kludgeins ...
 . Y321, Y322, Y323, Y325, Y378,
 Y379, Y425, Y426, Y506, Y522,
 Y523, Y529, Y530, Y531, Y540,
 Y556, Y560, Y570, Y1856, Y1887
 \@labels ...
 . J27, J146, J147, J189, J206, J207, 710
 \@largefloatcheck ...
 P192, P213, P238, P256, 1096
 \@lastchclass ...
 . L262, L272, L273, L275, L283,
 L308, L322, L326, L335, L348, L349
 \@latex@error h2836, l163, l217, l233,
 l239, l241, l244, l246, l248, l252,
 l254, l256, l259, l263, l268, l272,
 l274, l276, l277, l279, l282, l286,
 l288, o10, c100, r224, r268, r286,
 c159, c184, c195, s50, s84, v5, v28,
 v58, v102, v144, v187, v253, v339,
 x105, y100, y111, z23, z68, z97,
 z117, z170, z201, z231, z263, z364,
 z380, z478, z499, z551, z601, z605,
 z647, z652, z707, z775, z781, z825,
 z829, z833, z868, z872, z876, z933,
 z943, z1028, z1033, z1036, z1068,
 z1071, z1144, z1147, z1150, z1217,
 z1223, A49, A60, A82, A93, A600,
 A720, D143, H182, H237, H254,
 H287, H516, H531, H546, H558,
 I422, J219, L100, L109, e80, O31,
 P6, P83, R71, R88, U625, U676,
 U689, f128, U879, U927, U974,
 f155, U1065, U1082, U1090, U1095,
 U1117, U1174, U1252, U1307,

File Key: a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltruatex.dtx, e=lxpexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

U1643, U1673, f290, f329, f357,
 Y236, Y392, f481, f522, Y1878, Y1895
 \@latex@info . l163, l208, s85, E76,
 f236, W488, f310, f348, f376, f835, 334
 \@latex@info@no@line
 l163, l209, X89, Y580, 1100
 \@latex@note l190, v33
 \@latex@note@no@line
 l190, U1131, U1151, U1157, 1144
 \@latex@warning
 l163, l215, r344, r396, s55, G18,
 G35, G52, K350, M449, P260, R42,
 R69, R86, U1213, U1220, U1346,
 U1352, U1435, U1441, X174, Y2034
 \@latex@warning@no@line l163, l216,
 r19, r89, r146, r739, G8, G88, G89,
 H58, H65, H98, e98, O32, U291,
 U305, U704, U867, U1002, U1133,
 U1282, U1288, U1311, U1394,
 U1401, U1468, f202, U1545, X79,
 X98, X369, Y245, Y277, Y1831, Y2100
 \@latexbug l275, Y335, Y1817
 \@latexerr l215, Y439, 1097
 \@latexinfo 1095
 \@latexrelease@catcode@null g6, g2900
 \@lbibitem R3, R4
 \@ldots B504, B506
 \@leftcolumn
 Y123, Y2241, Y2262, Y2286, Y2295
 \@leftmark T16, T79
 \@let@token p411, p412,
 p419, D83, D96, I256, I258, I261,
 e49, e51, f782, f785, f788, f796, 1096
 \@align I208, I210
 \@linechar
 M190, M191, M192, M196,
 M197, M199, M204, M206, M207,
 M208, M209, M211, M215, M216,
 M219, M220, M225, M272, M670, 1126
 \@linefnt M124, M127, M190,
 M265, M273, M304, M307, M677
 \@linelen
 M164, M165, M176, M177, M203,
 M210, M219, M221, M226, M227,
 M228, M241, M242, M256, M257,
 M300, M303, M305, M306, M671
 \@list 711
 \@listctr J202, J225, R9, 710
 \@listdepth
 J23, J35, J38, J43, J99, K359, 710
 \@listfiles . r62, r131, r186, r745, r764
 \@listi 711
 \@listii 711
 \@listvi 711

\clnbk 1120
 \loadwithoptions
 U631, U653, U663, U672
 \lowpenalty p118, aa3
 \ltab L71, L106
 \ltxnomath 1095
 \cm p289, p430, p435, r45,
 r115, r173, J80, M213, M217, R37,
 b21, b429, b431, b432, b498, b499, 1112
 \mainaux r3, r37, r38, r107,
 r108, r165, r166, r217, r299, r342,
 r361, r394, r408, r433, H22, H82, 1106
 \makebox K11, K20, K25
 \makecaption P24
 \makecol Y263, Y415, Y462, Y484
 \makefcolumn Y395, Y396,
 Y404, Y406, Y442, Y444, Y452,
 Y454, Y2199, Y2201, Y2217, Y2218
 \makefnmark P400, P532
 \makefntext
 K392, K409, K426, P476, P494, P512
 \makeother a100, a129, v431, v432,
 v433, v434, v435, v436, v437, v438,
 v439, v440, v441, H438, H459,
 H552, H567, H577, U377, U378,
 U1195, U1328, U1417, f800, f801, a79
 \makepicbox
 K10, K19, K46, M366, M424
 \makespecialcolbox Y507, Y526, 1085
 \marbox P320, P322,
 P326, P330, P331, P379, Y1816,
 Y1826, Y1829, Y1837, Y1839,
 Y1840, Y1842, Y1843, Y1844, Y1853
 \marginparreset ... P343, P361, P370
 \markright T37, T62, T77
 \maxdepth r60, r129,
 r184, Y91, Y490, Y518, aa152, 1092
 \maxtab L2, L94, 1096
 \medpenalty p119, aa3
 \meta@family@list
 A116, A141, A153, A242, A506
 \midlist . Y66, Y503, Y504, Y1031,
 Y1033, Y1145, Y1309, Y1960, Y1988
 \minipage... 1118
 \minipagemode J181, K323,
 K325, K372, P187, P250, P345, P363
 \minipagerestore K360, K362
 \minipagetrue K324, P186
 \minus f26,
 Y2344, Y2345, Y2346, Y2349, Y2350
 \missing@onefilewithoptions ...
 U841, U897, U1015
 \missingfile@area
 r646, r687, r700, U899

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=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

```

\@missingfile@base ..... r646, r688, r701, U900
\@missingfile@ext r646, r689, r702, U901
\@missingfileerror ..... r641, r656, r666, r675, U898, U995, 398
\@mkboth ..... T11, T13
\@mklab ..... J45, J140
\@mkpream ..... L189, L234, L262, 1119
\@mparbottom ..... P387, P388, Y120, Y480,
Y1827, Y1835, Y1836, Y1837, Y1838
\@mpargs ..... K343, K375
\@mparswitchfalse ..... Y104
\@mpfn . K357, P452, P457, P541, P545
\@mpfootins ..... K349, K366, K367, K370, K376,
K383, K384, K401, K402, K418, K419
\@mpfootnotetext ..... K358, K378
\@mplistdepth ..... K359, K376
\@multicnt ..... L370, L372, L373,
L374, L381, L382, L383, M103,
M104, M106, M115, M116, M118,
M667, M704, M706, M707, M708,
M710, M711, M717, M723, M734,
M738, M764, M766, M768, M770,
M771, M775, M779, M790, M794, 1114
\@multiplelabels ..... r33, r103,
r161, G87, G93, H56, H62, H96, H102
\@multiput ..... M86, M95, M98
\@multispan ..... L371, L375, L379
\@mypkg@name ..... 925
\@mypkg@other@name ..... 925
\@namedef .. r446, v135, v136, v160,
w5, w525, x418, z401, z405, z414,
E57, E93, E869, G90, H215, H225,
H252, H278, H490, H499, I426,
I427, L175, N12, N13, N18, N19,
N23, N24, N25, f50, U834, U1108,
U1243, W483, aa480, aa481, 1084
\@nameuse r336, r392, r431, r445, N23,
T5, f51, U835, W487, Y611, Y669, 1100
\@nbitem ..... J168, J221
\@ne ..... b16, 1100
\@needsf@rmat ..... U699, U702, U707
\@needsformat ..... U687, U697, U701
\@negargfalse ..... M186
\@negargtrue ..... M185
\@newcommand ..... f79, f80
\@newctr ..... t13, t15, N8
\@newenv ..... f150, f151, f160
\@newenva ..... f149, f148
\@newenvb ..... f151, f150
\@newfontswitch ..... 1089
\@newl@bel .. G84, H24, H84, R10, 1112
\@newline ..... p95, p97
\@newlistfalse ..... J29, J33, J108, J182, Y604, Y662
\@newlisttrue ..... J29, J33, J87
\@next . P60, P129, P319, P320, Y9,
Y215, Y313, Y881, Y901, Y1816, b275
\@nextchar L269, L270, L330, L331, L332
\@nil ..... k13,
k19, k27, c13, c19, c83, c104, c105,
c117, c118, q14, r247, r248, r249,
c165, c166, c167, s89, s110, s996,
s1000, s1063, s1075, s1077, v383,
v394, v510, v525, v629, v632, v633,
v641, w431, w433, w465, w467,
w645, w647, w671, w673, x350,
x351, x353, x366, x372, x376, x377,
x413, x434, x439, x519, x533, y26,
y44, y53, y57, a164, z40, a165,
z571, z579, z612, z1228, z1230, D58,
D62, L367, L368, f53, f54, f58,
f63, U90, U91, U97, U105, U108,
U115, U140, U191, U192, U203,
U204, U214, U215, U217, U386,
U409, U452, U457, U573, U593,
U597, U603, U607, U795, U804,
f135, U821, U822, U1518, U1523,
U1526, U1528, U1529, U1544,
U1564, U1581, U1635, U1657,
U1681, W168, W176, W370, W412,
W414, f451, f452, f633, f776, f777
\@nbrlistfalse .. J33, J46, J91, 1082
\@nbrlisttrue ..... J225
\@nnil k13, k20, k21, k22, k28, r248,
r249, v214, v218, v219, v220, v235,
x179, x181, x345, x347, x359, x361,
x366, x380, x382, x389, x400, x401,
x403, x434, x439, M13, U740, U741,
U748, U768, U769, U776, f429, f435
\@no@font@optfalse ..... y17, y129
\@no@lnbk ..... p9, p10, p41, 1120
\@no@pgbk ..... p7, p8, p33, 1113
\@nobreak ..... 848
\@nobreak... ..... 1119
\@nobreakfalse ..... p121,
p123, J193, O94, O129, O157,
P182, Y167, Y194, Y1154, Y1320, 1119
\@nobreaktrue ..... p122, O126, P181
\@nocnterr ..... l240, 1093
\@nocnterr ..... l240, t4, t8, t16, t62, t64, N21, 1094
\@nodocument ..... l245, n136,
r68, r137, r192, r296, r358, H173,
P39, P108, Y158, Y185, Y214, 1119
\@noitemargfalse ..... J32, J200

```

File Key: a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\@noitemargtrue J32, J143
 \@noitemerr I278,
 p249, p267, p304, p327, J69, J81, J107
 \@noitemerror 1106
 \@noligs
 H439, H460, H568, H578, H589, 1087
 \@nolnbk 1120
 \@nolnerr
 . . . I238, p43, p114, H331, H338, 1096
 \@nomath v1, v337, A551,
 A586, A592, A613, A615, A637, 1080
 \@noparitemfalse J30, J145
 \@noparitemtrue J30, J66
 \@noparlistfalse J31, J70
 \@noparlisttrue J31, J67
 \@nopgbk 1113
 \@normalcr p53, p93, K322, 1136
 \@normalsize U4, U5, 1090
 \@normalsize_check 1093
 \@noskipsec 710
 \@noskipsecfalse
 . . . r54, r124, r179, O98, Y160, Y187
 \@noskipsectrue O38, O95
 \@notdefinable
 . . . I232, f136, f137, f141, f478, f519
 \@notprerr I281, r66, r135, r190
 \@nthm N3, N4
 \@nxttabmar L11, L21, L23,
 L25, L75, L111, L112, L118, L119, 1115
 \@obsoletefile r738
 \@oddfoot
 . . . T11, T14, T15, Y126, Y614, Y673
 \@oddhead T11, T14, Y125, Y614, Y673
 \@onefilewithoptions
 . . . U732, U736, U742, U760,
 U764, U770, U787, U791, U797,
 U813, U815, U895, U962, U1494, 940
 \@onefilewithoptions@clashchk
 . . . U830, U877
 \@onelevel@sanitize
 . . . P42, P111, f802, 1104
 \@onfilewithoptions 904
 \@onlypreamble h2719, r197,
 r206, r243, r740, r770, s23, s24, s61,
 s62, s66, s125, s145, s189, s190,
 s204, v17, v115, v117, v123, v139,
 v167, v182, v203, v208, v250, v537,
 x419, y28, y36, y42, y79, y83, y88,
 y93, y98, y108, y126, y127, y128,
 y134, y138, y142, z17, z19, z44,
 z46, z107, z116, z136, z394, z395,
 z408, z454, z502, z514, z516, z529,
 z554, z611, z613, z655, z694, z710,
 z787, z881, z890, z946, z949, z952,
 z972, z985, z1039, z1074, z1085,
 z1099, z1153, z1173, z1177, z1241,
 D140, D141, E58, E59, E60, E870,
 G92, Q12, Q29, R64, R81, S16,
 S272, U10, U13, f66, U85, U112,
 U120, U122, U154, U296, U359,
 U365, U430, U433, U434, U445,
 U446, U447, U472, U478, U491,
 U528, U546, U566, U586, U610,
 U615, U619, U622, U630, U651,
 U654, U664, U683, U696, U701,
 U707, U716, U721, U809, U895,
 U1020, U1029, U1037, U1038,
 U1056, U1063, U1072, U1079,
 U1080, U1088, U1093, U1098, f188,
 f190, U1474, f199, U1475, U1476,
 U1477, U1479, f207, V155, 1144
 \@opargbegintheorem N32, N35
 \@opcol Y264, Y272,
 Y396, Y415, Y444, Y462, Y467, 866
 \@options U565
 \@othm N3, N20
 \@outerparskip J8, J88, J117, J152, J222
 \@outputbox
 . . . S284, S287, S291, S315, S318,
 S322, Y122, Y487, Y489, Y509,
 Y512, Y513, Y533, Y535, Y536,
 Y541, Y544, Y549, Y551, Y558,
 Y564, Y566, Y640, Y699, Y727,
 Y733, Y743, Y744, Y767, Y774,
 Y860, Y863, Y866, Y872, Y873,
 Y2241, Y2245, Y2246, Y2260,
 Y2266, Y2286, Y2292, Y2301, 866
 \@outputdblcol
 . . . Y471, Y2236, Y2238, Y2282, Y2283
 \@outputpage . . . Y405, Y454, Y474,
 Y594, Y2270, Y2275, Y2308, Y2316
 \@oval M453, M471
 \@ovbtrue M476, M504, M534
 \@ovdx M431, M487, M489,
 M495, M497, M516, M518, M526,
 M528, M543, M551, M553, M589,
 M592, M599, M601, M693, M694,
 M695, M696, M712, M713, M714,
 M717, M733, M753, M754, M755,
 M756, M772, M773, M775, M789
 \@ovdy M431, M488, M490,
 M496, M497, M517, M519, M527,
 M528, M544, M552, M553, M564,
 M569, M577, M581, M700, M701,
 M702, M703, M718, M719, M720,
 M723, M737, M760, M761, M762,
 M763, M776, M777, M779, M793
 \@ovhlinefalse M477, M505

File Key: a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

\@ovhlinetrue M456, M460,
 M465, M483, M489, M511, M518
 \@ovhorz M494, M495,
 M525, M526, M550, M551, M584
 \@ovltrue M476, M504, M534
 \@ovri .. K33, M431, M486, M515,
 M542, M564, M577, M593, M602
 \@ovro
 M431, M486, M495, M496, M515,
 M526, M527, M542, M551, M552,
 M563, M569, M576, M581, M588,
 M598, M614, M621, M630, M639
 \@ovrtrue M476, M504, M534
 \@ovttrue M476, M504, M534
 \@ovvert M492, M493,
 M521, M523, M546, M548, M557
 \@ovvlinefalse M477, M505
 \@ovvlinetrue
 ... M459, M482, M490, M510, M519
 \@ovxx . M431, M480, M482, M483,
 M487, M493, M494, M508, M510,
 M511, M516, M523, M525, M537,
 M539, M543, M548, M550, M588,
 M598, M690, M691, M692, M696,
 M705, M706, M715, M716, M717,
 M732, M750, M751, M752, M756,
 M765, M766, M774, M775, M788
 \@ovyy . M431, M481, M482, M483,
 M488, M495, M509, M510, M511,
 M517, M526, M538, M539, M544,
 M551, M561, M574, M697, M698,
 M699, M703, M705, M721, M722,
 M723, M736, M757, M758, M759,
 M763, M765, M778, M779, M792
 \@p@filename
 . U90, U97, U105, U108, U115, U120
 \@p@filepath U91, U138, U147
 \@p@filepath@aux . U139, U140, U148
 \@pagedp Y119, Y310,
 Y315, Y1095, Y1248, Y1845, Y1855
 \@pageht Y118,
 Y311, Y315, Y317, Y318, Y319,
 Y323, Y1094, Y1247, Y1828, Y1835
 \@par m3, m5
 \@parboxrestore K266, K322,
 K356, K387, K405, K422, P19,
 P100, P169, P342, P360, P470,
 P489, P507, Y221, Y605, Y663, 1119
 \@parboxto K261, 1127
 \@parmmoderr l271, P58, P127, P316
 \@parse@version c83,
 c104, c105, c117, c118, c165, c166,
 c167, U197, U203, U204, U214,
 U1529, U1544, U1581, U1657, U1681
 \@parse@version@ .. U191, U192, U197, U209
 \@partaux
 ... r3, r223, r267, r285, r309, r311,
 r312, r332, r371, r373, r374, r388,
 r418, r420, r421, r427, r439, r441, r444
 \@partlist r232, r236,
 r237, r239, r263, r282, r305, r367, r414
 \@partswfalse r6
 \@partswtrue r231, r261, r281
 \@pass@ptions U382, U384,
 U404, U405, U407, U420, U421,
 U423, U430, U431, U432, U906, U988
 \@pboxswfalse K264, K341
 \@pboxswtrue K274
 \@pdef 1089
 \@openup I199, I200
 \@percentchar a109,
 U1187, U1189, U1191, U1193,
 U1320, U1322, U1324, U1326,
 U1409, U1411, U1413, U1415, U1463
 \@pgbk 1113
 \@picbox M6, M31, M43, M51, M52, 1081
 \@picht M6, M29, M42, M51
 \@picture M23, M24
 \@picture@warn
 ... M223, M441, M445, M449
 \@pkgetension 1089
 \@pkgextension U31, U155,
 U163, U221, U431, U621, U624,
 U663, U672, U743, U771, U798,
 U917, U944, U1074, W483, W493, 902
 \@plus p458, O16, O206,
 O229, f26, T83, Y2344, Y2345,
 Y2346, Y2349, Y2350, Y2354,
 Y2355, Y2356, Y2360, Y2361, Y2362
 \@pnumwidth O218, O241
 \@popfilename . U33, U875, U1010, 880
 \@pr@videopackage U295, U309,
 U312, U333, U335, U346, U348, U359
 \@preamble L190,
 L192, L200, L237, L256, L258,
 L259, L263, L278, L296, L297, L334
 \@preamblecmds r67,
 r136, r191, f66, U1695, U1696, 1089
 \@preamerr l260, L199, L274, L355
 \@process@pti@ns U490, U508,
 U525, U532, U533, U546, U550, U552
 \@process@ptions .. U477, U479, U491
 \@protect@... 1100
 \@protected@testopt f89, f101, f628, f640
 \@protecteddef 1095
 \@providesfile a101, a102, U369, aa709

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

\@optionlist [U152](#), [U224](#),
 [U476](#), [U883](#), [U889](#), [U978](#), [U984](#), [U1075](#)
 \@pushfilename . [U33](#), [U823](#), [U965](#), [292](#)
 \@put [M452](#),
 [M497](#), [M528](#), [M553](#), [M621](#), [M639](#)
 \@qend [l236](#), [f136](#), [f776](#)
 \@qrelax [f137](#), [f776](#)
 \@raw@classoptionslist
 [U11](#), [U730](#), [V64](#), [V104](#)
 \@rc@ifdefinable . [s14](#), [f130](#), [f132](#), [f250](#)
 \@reargdef [f122](#), [1089](#)
 \@refundefined
 ... [r55](#), [r125](#), [r180](#), [G3](#), [H54](#), [H94](#), [1115](#)
 \@reinserts [Y329](#), [Y332](#), [Y520](#)
 \@remove@eq@value . [U448](#), [U513](#), [U573](#)
 \@remove@tlig [s996](#), [s1004](#)
 \@remove@tlig@ [s996](#), [s997](#)
 \@remove@tlig@@ [s997](#), [s1000](#)
 \@remove@tlig@@@ [s1010](#), [s1029](#)
 \@removeelement [k32](#), [U572](#), [U581](#)
 \@removefromreset
 [t46](#), [t70](#), [t71](#), [t81](#), [t104](#), [t107](#)
 \@renewfontswitch [1092](#)
 \@reqcolroom [Y1094](#),
 [Y1095](#), [Y1098](#), [Y1100](#), [Y1101](#),
 [Y1106](#), [Y1110](#), [Y1112](#), [Y1140](#),
 [Y1141](#), [Y1247](#), [Y1248](#), [Y1252](#),
 [Y1255](#), [Y1256](#), [Y1261](#), [Y1268](#),
 [Y1270](#), [Y1302](#), [Y1303](#), [Y1414](#),
 [Y1416](#), [Y1418](#), [Y1421](#), [Y1423](#),
 [Y1497](#), [Y1500](#), [Y1503](#), [Y1508](#),
 [Y1510](#), [Y2024](#), [Y2141](#), [Y2146](#), [Y2149](#)
 \@reserveda [498](#)
 \@reset@ptions [U839](#),
 [U876](#), [U904](#), [U969](#), [U1011](#), [U1021](#)
 \@resetactivechars [Y579](#), [Y602](#), [Y660](#)
 \@resethfps [Y1209](#), [Y1378](#), [Y2091](#), [1096](#)
 \@resetprotect [1110](#)
 \@restorepar [m5](#), [p342](#),
 [p358](#), [p376](#), [p392](#), [J127](#), [J135](#), [1112](#)
 \@reversemarginfalse ... [P388](#), [Y103](#)
 \@reversemargintrue [P387](#)
 \@rightmark [T16](#), [T80](#)
 \@rightskip
 ... [H370](#), [H389](#), [H407](#), [J75](#), [K295](#), [K316](#)
 \@rjfieldfalse [L34](#), [L77](#)
 \@rjfieldtrue [L125](#)
 \@rmfamilyhook [A429](#), [A451](#), [A463](#), [A490](#)
 \@roman [t137](#), [t143](#)
 \@rsbox [K465](#), [K472](#), [K476](#)
 \@rtab [L71](#), [L86](#)
 \@rule [K437](#), [K442](#), [K446](#)
 \@sanitize ... [Q7](#), [Q18](#), [Q24](#), [Q35](#), [f800](#)
 \@savebox [K122](#), [K129](#), [K135](#)
 \@savemarbox . [P326](#), [P327](#), [P330](#), [P333](#)
 \@savepicbox [K122](#), [K129](#), [K139](#)
 \@savsf [o24](#), [o26](#), [p124](#), [p130](#),
 [p139](#), [p158](#), [p172](#), [p184](#), [p198](#), [p212](#)
 \@savsk [p124](#), [p129](#),
 [p140](#), [p159](#), [p173](#), [p185](#), [p199](#), [p213](#)
 \@scolelt [Y792](#), [Y857](#)
 \@sdblcolelt [Y809](#), [Y829](#), [Y858](#)
 \@secCntformat [O60](#), [O111](#), [1083](#)
 \@secondoffive .. [G24](#), [G27](#), [G41](#), [G58](#)
 \@secondoftwo
 ... [a91](#), [o5](#), [c87](#), [r509](#), [r539](#), [c185](#),
 [r585](#), [r602](#), [s131](#), [t190](#), [t195](#), [A512](#),
 [E71](#), [E838](#), [E889](#), [E905](#), [G44](#), [G80](#),
 [e36](#), [e40](#), [T17](#), [U159](#), [U193](#), [U205](#),
 [U238](#), [U256](#), [f212](#), [U1686](#), [f457](#),
 [f586](#), [aa715](#), [f637](#), [f745](#), [f752](#), [f772](#), [f819](#)
 \@secPenalty [p18](#), [O36](#), [O50](#)
 \@sect [O54](#), [O55](#)
 \@seqncr [I425](#)
 \@set@curr@file@aux [W364](#), [1150](#)
 \@setckpt .. [r439](#), [r446](#), [H23](#), [H83](#), [1083](#)
 \@setfloattypecounts [Y1078](#), [Y1228](#),
 [Y1399](#), [Y1480](#), [Y1575](#), [Y1696](#), [Y2038](#)
 \@setfontsize [A637](#)
 \@setfps [P34](#)
 \@setfpsbit [P73](#), [P75](#),
 [P77](#), [P85](#), [P143](#), [P146](#), [P149](#), [Y2082](#)
 \@setmarks [Y2252](#), [Y2254](#), [Y2269](#)
 \@setminipage
 ... [K361](#), [P21](#), [P177](#), [P185](#), [P376](#), [1107](#)
 \@setminipage [1120](#)
 \@setnobreak [P179](#), [P375](#), [1107](#)
 \@setpar [m3](#), [J78](#)
 \@setprotect [1110](#)
 \@setref [G10](#)
 \@setsize [A637](#)
 \@settab [L71](#), [L93](#)
 \@settodim [u17](#)
 \@settopoint [u22](#)
 \@setupverbvisiblespace
 ... [H480](#), [H491](#), [H503](#), [H520](#), [H535](#)
 \@sffamilyhook [A429](#), [A456](#), [A464](#), [A491](#)
 \@sharp [L196](#), [L235](#), [L265](#),
 [L280](#), [L281](#), [L301](#), [L303](#), [L305](#), [L333](#)
 \@shipoutsetup [Y594](#)
 \@shortstack [M132](#), [M133](#)
 \@show@ [100](#)
 \@show@DeclareRobustCommand
 ... [f554](#), [f568](#), [f613](#), [f667](#), [f670](#)
 \@show@DeclareRobustCommand@env
 ... [f691](#), [f693](#)
 \@show@environment [f685](#), [f686](#)

File Key: a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=lptra.dtx, o=ltmeta.dtx, p=ltspac.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\@showenvironment@begin f694, f700, f708, f713
 \@showenvironment@end f698, f707, f712, 105
 \@showenvironment@end@aux f716, f718
 \@showmacro f613, f703
 \@shownewcommand f555, f568, f621, f648, f667, f673, 102
 \@shownewcommand@aux f648, f677, f695, 103
 \@shownewcommand@env f693, f692
 \@shownonstop f703, f724, 105
 \@shownormalenv f689, f712
 \@showtokens ... f653, f658, f702, f719
 \@showtypeout ... f697, f702, f724, 105
 \@showcommandlisthook f551, f553, f561, g1318, 100
 \@showenvironmentlisthook f688, f690, g1320, 1150
 \@skipping@modulefalse c154, c173, c200
 \@skipping@moduletrue ... c153, c177
 \@sline M167, M179, M184, M269
 \@slowromancap t144, t145
 \@spaces l218
 \@specialoutput Y258
 \@specialpagefalse . Y99, Y611, Y669
 \@specialpagetrue T9
 \@specialstyle T9, Y611, Y669
 \@spoken f785, f795
 \@sqrt I355
 \@ssect O53, O112
 \@stackcr M139, M142
 \@star@or@long f72, f77, f124, f146, f152, f178, f187, f233
 \@startcolumn Y265, Y272, Y779
 \@startdblcolumn Y779, Y2274, Y2277, Y2314, Y2320
 \@startfield L28, L46, L92, L104, L125, L133
 \@startline L20, L57, L62, L68, L83, L154
 \@startpagehook 1086
 \@startpbox L193, L236, L266, L332, L384, L386
 \@startsection O39
 \@starttoc O149
 \@stopfield L32, L48, L86, L93, L125, L127, L140, L142, L154
 \@stopline L30, L56, L85
 \@stpelt t20, t23
 \@string@makeletter f806, 109
 \@strip@args s76
 \@strip@tex@ext r227, r237, r239, r244, r274, 388
 \@strip@tex@ext@aux r244, r275
 \@svector M244, M259, M269
 \@sverb . H507, H570, H579, H582, 689
 \@svsec ... O57, O60, O66, O78, 1083
 \@svsechd O76, O101, O121
 \@swaptwoargs r619, r621, U849, W153, W167, W187
 \@sxverbatim H415, H492, H499
 \@tabacckludge s223, s225, s454, s455, E193, E200, E202, 1115
 \@tabacol L178, L258
 \@tabarray L170, L180, L181
 \@tabclassiv L180, L330
 \@tabclassz L179, L282
 \@tabcr L56, L73
 \@tabfbox L16, L80, L82
 \@tablab L72, L126
 \@tabminus L72, L117
 \@tabplus L72, L110
 \@tabpush L11, L77, L85, L140, L143, L145
 \@tabrj L72, L124
 \@tabular L174, L177, L178
 \@tabularcr L180, L208
 \@tempa G120, G121, G122, G123, G147, G148, G158, G159, 1107
 \@tempboxa j13, s69, u17, u18, J205, J211, J212, J214, K29, K30, K31, K32, K37, K38, K39, K40, K175, K205, K212, K222, K344, K375, K482, K483, K484, K491, K492, K493, K494, M304, M305, M447, M448, M486, M491, M496, M497, M515, M520, M527, M528, M542, M545, M552, M553, M614, M615, M620, M621, M630, M631, M638, M639, M724, M742, M780, M798, O138, O139, P322, P380, Y307, Y379, Y384, Y385, Y426, Y431, Y432, Y570, Y630, Y637, Y638, Y689, Y696, Y697, Y725, Y729, Y741, Y747, Y754, Y755, Y756, Y757, Y761, Y769, 1082
 \@tempcnta j7, z954, z955, z956, z957, z961, L242, L243, L244, L245, M187, M188, M214, M215, M216, M229, M230, M231, M232, M239, M240, M254, M255, M270, M271, M276, M278, M279, M280, M281, M282, M285, M287, M288, M289, M290, M291, M292, M293, M294, M295, M296, M335, M336, M337, M338, M339, M360, M361, M362, M363, M364,

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=ltCMDBHOOKS.dtx, j=ltALLOC.dtx, k=ltCNTRL.dtx, l=ltERROR.dtx, m=ltPAR.dtx, n=ltPARA.dtx, o=ltMETA.dtx, p=ltSPACE.dtx, q=ltLOGOS.dtx, r=ltFILES.dtx, s=ltOUTENC.dtx, t=ltCOUNTS.dtx, u=ltLENGTH.dtx, v=ltFSSBAS.dtx, w=ltFSSAXES.dtx, x=ltFSSTRC.dtx, y=ltFSSCMP.dtx, z=ltFSSDCL.dtx, A=ltFSSINI.dtx, B=fontdef.dtx, C=preload.dtx, D=ltFNTCMD.dtx, E=ltTEXTCOMP.dtx, F=ltPAGENO.dtx, G=ltXREF.dtx, H=ltMISCEN.dtx, I=ltMATH.dtx, J=ltLISTS.dtx, K=ltBOXES.dtx, L=ltTAB.dtx, M=ltPICTUR.dtx, N=ltTHM.dtx, O=ltSECT.dtx, P=ltFLOAT.dtx, Q=ltIDXGLO.dtx, R=ltBIBL.dtx, S=ltMARKS.dtx, T=ltPAGE.dtx, U=ltCLASS.dtx, V=ltKEYS.dtx, W=ltFILEHOOK.dtx, X=ltSHIPOUT.dtx, Y=ltOUTPUT.dtx, Z=ltHYPHEN.dtx, aa=ltFINAL.dtx

M365, M392, M393, M394, M395,
 M396, M414, M416, M418, M419,
 M421, M423, M438, M439, M440,
 M442, M444, M446, M448, M562,
 M567, M575, M579, M616, M617,
 M618, M619, M632, M633, M635,
 M636, M658, M659, M660, M661,
 M662, M663, M711, M731, M771,
 M787, P62, P68, P70, P79, P80,
 P90, P91, P131, P137, P139, P152,
 P153, P159, P160, U1135, U1137,
 U1138, U1139, U1266, U1268,
 U1269, U1270, U1380, U1382,
 U1383, U1384, Y16, Y18, Y20,
 Y938, Y939, Y940, Y941, Y961,
 Y962, Y963, Y964, Y986, Y989,
 Y1022, Y1025, Y1136, Y1298,
 Y1578, Y1581, Y1699, Y1702,
 Y1817, Y1819, Y1822, Y1824,
 Y1826, Y1848, Y2072, Y2073,
 Y2077, Y2083, Y2087, aa227,
 aa232, aa233, aa234, aa364, aa366,
 aa367, aa368, aa375, aa377, aa378,
 aa379, aa385, aa387, aa388, aa389,
 aa396, aa398, aa399, aa400, aa426,
 aa428, aa429, aa430, aa452, aa454,
 aa455, aa456, aa462, aa464, aa465,
 aa466, aa495, aa500, aa501, aa502
 \tempcntb j7, z955, z959,
 z961, M279, M280, M281, M283,
 M284, M285, M562, M563, M567,
 M568, M575, M576, M579, M580,
 P88, P89, P90, P157, P158, P159,
 Y17, Y20, Y21, Y2083, Y2084,
 Y2085, aa228, aa232, aa496, aa500
 \tempdima . . . j10, v219, v224, I186,
 I189, I195, K43, K44, K204, K205,
 K210, K211, K212, K214, K265,
 K266, K342, K346, K449, K452,
 K453, K480, K482, K488, K491,
 L35, L36, L37, L88, L89, L90, L91,
 L218, L219, M210, M211, M213,
 M214, M215, M216, M217, M218,
 M437, M438, M439, M448, M487,
 M488, M492, M493, M516, M517,
 M521, M523, M543, M544, M546,
 M548, M617, M619, M634, M636,
 M637, M657, M658, M659, O211,
 O212, O234, O235, O248, P196,
 P198, P218, P220, P258, P259,
 P260, Y231, Y232, Y233, Y491,
 Y493, Y539, Y541, Y542, Y547,
 Y552, Y556, Y561, Y565, Y921,
 Y924, Y944, Y954, Y966, Y976,
 Y1641, Y1642, Y1645, Y1646,
 Y1766, Y1767, Y1771, Y1772,
 Y1827, Y1828, Y1829, Y1830,
 Y1833, Y1836, Y1839, Y1841,
 Y2190, Y2191, Y2193, Y2194, 1083
 \tempdimb j10, v220,
 v225, v649, v653, x179, x180, x437,
 x460, x461, x470, x471, x475, x493,
 x496, x499, x501, K268, K269,
 K450, K453, K481, K483, K489,
 K492, M211, M212, M357, M359,
 M362, M365, M482, M484, M485,
 M510, M513, M514, M539, M540,
 M541, M612, M613, M622, M628,
 M629, M640, M648, M649, M654,
 Y944, Y945, Y946, Y947, Y954,
 Y966, Y967, Y968, Y969, Y976, 1083
 \tempdimc j10, x454, x455,
 x457, x458, x460, x461, K53, K54,
 K64, K65, K451, K452, K453, M30,
 M31, M32, M33, M34, M35, M60,
 M61, M63, M64, M332, M333, M334
 \tempdimx 1083
 \tempskipa j14, p45,
 p48, p49, p286, p293, p295, p298,
 x181, x182, J116, J117, J118, J150,
 J152, J153, J154, J222, J223, J224,
 O42, O44, O45, O50, O62, O63,
 O88, O89, O91, O103, O104, O113,
 O114, Y1876, Y1877, Y1879, Y1887
 \tempskipb
 . . . j14, p221, p223, p225, p228,
 p230, p244, p262, p284, p286, p287,
 p291, p293, p295, p296, p319, p322, 372
 \tempswa 1079
 \tempswafalse
 . . . r303, r365, r412, s1537, v94,
 v698, z466, z541, z615, z696, z1216,
 z1222, H25, H85, H430, H451,
 R17, R29, U1108, U1124, U1243,
 U1259, W287, W310, W331, Y992,
 Y1028, Y1584, Y1705, b262, a81
 \tempswatrue
 . . . r301, r306, r363, r368, r410,
 r415, s1540, s1541, v97, v699, v700,
 v703, v706, z469, z544, z618, z699,
 z1179, H128, H435, H456, R17,
 R29, U1105, U1240, W287, W310,
 W331, Y1586, Y1609, Y1707,
 Y1732, Y2151, Y2168, b268, a82
 \temptokena j16, H140, H144, H153,
 H166, H167, T27, T31, T38, T42,
 T55, T56, T63, T64, T77, T78, 676
 \test 1118

File Key: a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\@test@opt 1114
 \@testdef H24, H84, H126
 \@testfalse Y12, Y14, Y15
 \@testfp Y886, Y906, Y942,
 Y965, Y2075, Y2204, Y2221, 1106
 \@testopt p7, p8, p9,
 p10, I396, f33, f79, f99, f103, f148, 1114
 \@testpach L270, L348
 \@testpatch L348
 \@testtrue Y13, Y21,
 Y358, Y889, Y908, Y948, Y970, Y2079
 \@testwrongwidth Y347,
 Y887, Y943, Y1116, Y1430, Y1635
 \@text@composite s76, s1070, s1075, 1108
 \@text@composite@x s76
 \@textbottom
 T83, T85, Y515, Y553, Y567, Y576
 \@textfloatsheight
 Y480, Y1091, Y1093,
 Y1143, Y1144, Y1149, Y1244,
 Y1246, Y1306, Y1308, Y1314, Y2024
 \@textmin P285, P286,
 P299, P300, Y114, Y1093, Y1097,
 Y1100, Y1101, Y1246, Y1251,
 Y1255, Y1256, Y1418, Y1503,
 Y1602, Y1604, Y1620, Y1724,
 Y1726, Y1744, Y2132, Y2134, Y2136
 \@textsubscript P424, P432
 \@textsupserscript
 P401, P403, P404, 1111
 \@texttop . T83, T85, Y511, Y534, Y576
 \@tf@r k25, k26, 1092
 \@tfor k25,
 r586, r603, r747, D88, K57, K76,
 L268, M478, M506, M535, P63, P132
 \@tforloop k27, k28, k30
 \@thanks O11, O34
 \@thefnmark K390, K407, K424, P400,
 P401, P453, P458, P473, P491,
 P509, P521, P526, P537, P542, P553
 \@thefoot Y126,
 Y614, Y617, Y644, Y673, Y676, Y703
 \@thehead Y125,
 Y614, Y616, Y634, Y673, Y675, Y693
 \@themargin Y74,
 Y615, Y617, Y629, Y674, Y676, Y688
 \@themark T26, T27, T37, T38,
 T41, T54, T55, T62, T63, T78, T81
 \@thirdofthree s197, f216
 \@thm N12, N18, N24, N26
 \@thmcounter N11, N17, N33
 \@thmcountersep N10, N33
 \@title O7, O31
 \@tocrmarg O207, O230

\@toodeep l253, J36, J232, J243
 \@toplist Y64, Y386, Y387,
 Y433, Y434, Y720, Y726, Y736,
 Y737, Y1029, Y1041, Y1958, Y1986
 \@topnewpage Y201, 1095
 \@topnum P271, Y107, Y1026, Y1027,
 Y1041, Y1045, Y1462, Y1467,
 Y1555, Y1562, Y1949, Y1977, Y2018
 \@toproom P273,
 Y108, Y1029, Y1041, Y1950, Y1978
 \@topsep J1, J71, J73, J171
 \@topsepadd ... J1, J59, J61, J71, J124
 \@totallftmargin
 H426, H448, J9, J53,
 J54, K294, K315, L35, L76, L81, 710
 \@trivlist J48, J57, J92
 \@tryfcolumn Y782,
 Y802, Y820, Y836, Y2205, Y2222
 \@trylist Y845, Y848, Y881, Y901, Y923
 \@ttfamilyhook A429, A461, A465, A492
 \@twoasseserror U612, U1094
 \@twocolumnfalse Y101, Y149
 \@twocolumntrue Y208
 \@twoloadclasserror
 U874, U1009, U1089
 \@twosidefalse Y102
 \@typein f32, f33, f40, f48, 1114
 \@typeset@protect .. s26, s32, s210,
 s218, A638, H200, H245, H263,
 f102, f255, f262, f264, aa358, 1106
 \@uclclist
 s1506, s1507, s1560, aa576, 1121
 \@undefined
 M57, M71, M82, M91, M162,
 M174, M237, M252, M313, M371
 \@undefined
 b582, b650, b690, b705, l28,
 a111, a112, c53, a113, n163, n164,
 c62, n165, n166, n167, o34, p90,
 p472, p478, r61, r62, r130, r131,
 r185, r186, r274, r275, r477, r478,
 r479, r484, r538, r551, r560, r575,
 r630, s195, s197, s333, s335, s337,
 s339, s341, s343, s345, s347, s349,
 s351, s370, s372, s374, s458, d2,
 s700, s703, d15, d16, d17, d29, d30,
 d74, a134, d84, t204, a142, d173,
 v459, v499, v561, v594, d189, v658,
 v665, d197, d205, a150, d237, d238,
 d239, d240, d241, d242, w388, d243,
 d244, d245, w411, d246, w420,
 d247, d248, d249, d250, d251, d252,
 d253, d254, d255, w509, w510,
 w511, w512, w513, w514, w515,

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

w516, w517, w518, d261, w529,
 w534, w539, w603, w604, w605,
 w606, w607, w608, w609, w634,
 w714, w716, w717, w719, w720,
 w721, w723, a157, x552, x553, y4,
 y5, y6, y7, y8, y9, y10, y11, y12,
 y13, y14, y15, y16, y17, y18, y19,
 y20, z141, z509, z561, z812, z919,
 A35, A58, A71, A91, A104, A105,
 A106, A107, A108, A109, A110,
 A111, A112, A113, A114, A116,
 A117, A118, A231, A232, A233,
 A234, A235, A236, A237, A238,
 A239, A240, A242, A243, A244,
 A304, A305, A472, A490, A491,
 A492, A534, A535, A536, A537,
 A538, A578, A579, A580, A581,
 A582, A583, A594, A716, B15, B50,
 B65, D37, D38, D39, D122, E159,
 E367, E617, E620, E621, E651,
 E652, E653, E656, E657, E658,
 E659, E660, E661, E666, E667,
 E668, E669, E674, E675, E676,
 E677, E680, E681, E682, E684,
 E685, E690, E691, E692, E693,
 E694, E695, E696, E697, E698,
 E699, E700, E701, E702, E703,
 E704, E705, E707, E708, E710,
 E711, E712, E713, E714, E715,
 E716, E717, E718, E719, E721,
 E722, E723, E724, E725, E726,
 E727, E728, E729, E731, E732,
 E733, E734, E735, E736, E737,
 E738, E739, E740, E741, E742,
 E743, E744, E745, E746, E747,
 E748, E749, E750, E751, E752,
 E753, E754, E755, E756, E757,
 E762, E763, E765, E766, E767,
 E768, E769, E770, E771, E772,
 E774, E775, E777, E778, E780,
 E781, E782, E783, E785, E786,
 E787, E789, E791, E792, E793,
 E794, E795, E796, E797, E799,
 E800, E801, E802, E803, E804,
 E805, E806, E807, G130, G131,
 G132, G152, a208, G168, G169,
 H108, H160, H161, H162, H163,
 H298, H299, H319, H320, H321,
 H322, a212, H474, H501, H502,
 H503, H504, H537, I236, I246, I247,
 I283, I286, I329, I342, I436, K21,
 K130, K194, K250, K443, e4, K473,
 e13, e14, e15, e16, M18, a238,
 M467, M468, e147, e148, e149,
 e150, e151, e152, O197, O245, P5,
 a245, e169, e170, e171, e172, e192,
 e193, P429, P448, P562, R30, R53,
 f34, U4, U25, U146, U147, U148,
 U209, U355, U463, U513, U896,
 U934, U936, U942, U999, U1014,
 U1015, U1030, U1148, U1226,
 U1229, U1279, U1358, U1361,
 U1371, U1372, U1373, U1374,
 U1375, U1376, U1377, U1448,
 U1451, U1467, U1487, U1494, f235,
 W18, W19, W20, W21, W147,
 W189, W190, W197, W198, W199,
 W321, W342, W356, W360, W361,
 W493, W499, W500, W501, f319,
 f320, f351, X438, X439, X440,
 X441, X442, X444, X445, X446,
 X453, X454, X456, X458, X459,
 X460, X463, X486, f379, Y36,
 f383, f384, f401, Y370, Y371, f415,
 f462, f463, f464, f465, f466, f467,
 f468, f504, f505, f506, f507, f508,
 f509, Y1928, Y1929, Y1930, Y1931,
 f540, f541, f542, f560, f561, aa10,
 aa18, aa39, aa54, aa73, aa82, aa89,
 aa98, aa108, aa160, aa161, aa270,
 aa283, aa296, aa316, aa341, aa351,
 aa352, aa353, aa382, aa384, aa423,
 aa424, aa443, aa444, aa445, aa446,
 aa447, aa448, aa449, aa450, aa451,
 aa468, aa484, aa485, aa486, aa534,
 aa535, aa663, aa698, aa699, aa700,
 aa701, aa702, a329, a330, f668,
 f669, f670, f671, f672, f673, f675,
 f676, f677, f732, f747, f754, f828,
 f829, f830, f839, b65, b81, b105,
 b106, b121, b122, b127, b136,
 b149, b184, b189, b222, b223, b246,
 b256, b291, b353, b363, b365, a71,
 a72, b475, b476, b486, b487, 1133
 \undefinedfonterror 1096
 \unexpandable@protect
 r211, L264, f232, f267, f273, f278, 1105
 \unknownonoptionerror
 U1025, U1064, U1077
 \unknownversion 1085
 \unprocessedoptions
 U562, U671, U896, U935,
 U936, U996, U1000, U1079, V52, 903
 \unused 115,
 l32, l59, f10, f17, U1472, b307, 1143
 \unusedoptionlist
 ... r18, r20, r88, r90, r145, r147,
 U12, U455, U456, U466, U467,

File Key: a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

U574, U582, V95, V98, V99, V116, 929
 \@upline M297, M298, M304
 \@upordown M195, M196, M204, M225, M273
 \@upvector M268, M304
 \@use@option U486, U502, U520, U536, U538, U555, U557, U567
 \@use@text@encoding s150, E16, E1141
 \@vbsphack p220
 \@verb H570, H579, H582
 \@verbatim ... H420, H466, H490, H499
 \@verbvisiblespacebox H485, H486, H489, H504
 \@vereq B457, B458
 \@viiipt v741
 \@viipt v740
 \@vipt v739
 \@vline M166, M178, M297
 \@vobeyspaces H413, H466, H492, H520, H535, H582
 \@vpt v738
 \@vspace p331
 \@vspace@calcify p80, p102, p242, p338, p343, p353, p361, H349, I404, L62, L224, M148
 \@vspacer p331
 \@vtryfc Y851, Y859
 \@vvector M243, M258, M268
 \@warning l215
 \@wckptelt r440, r443
 \@whiledim k7, M123, M203
 \@whilenoop k3, 1113
 \@whilenum .. k3, L244, M104, M116, M336, M338, M361, M364, M393, M395, M416, M421, M731, M787
 \@whilesw k10, Y266, Y396, Y405, Y443, Y453, Y2275, Y2315
 \@whilesnoop k10, 1113
 \@wholewidth K160, K162, K163, K165, K167, K168, K169, K170, M2, M126, M129, M131, M299, M302, M350, M359, M406, M413, M565, M578, M590, M600, M679, M680, M728, M784
 \@width p454, s289, s294, x192, B618, K165, K167, K218, K225, K453, K497, L188, L219, L347, L366, M227, M299, M302, M325, M333, M350, M359, M384, M392, M406, M413, M565, M578, M728, M784, P395, f26, Y1855, Y2264, Y2298, b505, 1109
 \@wrglossary Q25, Q30, 1105
 \@wrindex Q8, Q13, 1105
 \@writeckpt r330, r386, r425, r437
 \@writefile r32, r102, r160, H147, O183, 676
 \@writesetup Y594
 \@wrong@font@char s161, v562, v596, v609, 1116
 \@wtryfc Y861, Y871
 \@x@protect f105, f254
 \@x@sf P531, P533
 \@xDeclareMathDelimiter z984, z1040
 \@xaddvskip p220, p245, p263
 \@xarg M163, M166, M175, M178, M185, M189, M190, M226, M228, M238, M239, M243, M253, M254, M258, M266, M274, M664
 \@xargarraycr L205, L214, L218
 \@xargdef f80
 \@xarraycr L202, L203
 \@xbitor Y15, Y17
 \@xcentercr H332, H339, H343
 \@xdblarg f798
 \@xdblfloor P264, 1104
 \@xdim M84, M93, M105, M107, M117, M119, M668, M732, M733, M734, M735, M741, M788, M789, M790, M791, M797
 \@xeqnacr I388
 \@xexnoop L238, L248
 \@xexpast L239, L240
 \@xfloat .. P28, P29, P34, P266, 1105
 \@xfootnote P452, P455, 1082
 \@xfootnotemark ... P519, P523, 1082
 \@xfootnotenext P536, P539
 \@xfootnotetext 1082
 \@xhline L360, L361
 \@xifnch f786, f796
 \@xipt v745, B164, B166, B167
 \@xipt v744, B163
 \@xivpt v746, B165, B167
 \@xmpar P324, P325
 \@xnewline p61, p62, p73, p74, p94
 \@xnext Y10, Y11
 \@xnthm N5, N6
 \@xobeysp p421, H414, H415, H482, H486, 376
 \@xprocess@options U477, U494, U496, U512, U514, U528
 \@xpt v743, B162, B165, B166
 \@xsect O86, O87, O123
 \@xtabcr L56, L57
 \@xtabularcr L209, L210
 \@xthm N28, N29
 \@xtryfc Y848, Y876
 \@xtypein f33, f35, f42

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfnctcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\@xverbatim H415, H466
 \@xviipt v747, B166, B168
 \@xxDeclarMathDelimiter . z969, z973
 \@xxpt v748, B167, B168
 \@xxvpt v749, B168
 \@xxxii j2, s425, s427, P89,
 P158, Y883, Y884, Y903, Y904,
 Y939, Y940, Y962, Y963, Y2041, 1116
 \@xympar P328, P332, P378
 \@yarg
 M163, M167, M175, M179, M185,
 M186, M195, M238, M244, M253,
 M259, M268, M270, M297, M664
 \@yargarraycr L206, L216, L220
 \@yargd@f f107
 \@yargdef f84, f94, f107, f123, 1118
 \@ydim M85, M94,
 M105, M108, M117, M119, M668,
 M736, M737, M738, M739, M740,
 M792, M793, M794, M795, M796
 \@yeqncr I388
 \@ympar P324, P329
 \@ynthm N5, N14
 \@ythm N28, N29
 \@tryfc Y894, Y913, Y917
 \@yyarg M185,
 M186, M187, M190, M274, M664
 \@ztryfc Y922, Y933
 \@zend 217
 \accent@spacefactor s70, s73, s74, 1124
 \active@math@prime
 I253, I254, Y592, 1124
 \add@accent s65, s67, 1117
 \add@percent@to@temptokena
 H134, H150, H152, H161, 677
 \add@unicode@accent s1039, s1053
 \addto@hook
 ... v152, v154, v737, z438, z574,
 z578, z595, z719, z725, z733, z749,
 z752, z755, z1188, z1195, z1198, 1091
 \AddToHook 298
 \AddToHookNext 298
 \AddToHookNextWithArguments ... 298
 \AddToHookWithArguments 298
 \alloc@ d22, d26,
 d38, v14, b90, b91, b92, b93, b94,
 b95, b96, b97, b98, b99, b226, 1139
 \alpha@elt
 ... z45, z442, z669, z771, z1187, z1188
 \alpha@list
 ... z41, z43, z451, z657, z669,
 z714, z769, z770, z1183, z1189, z1190
 \apptocmd 299
 \atveryend@DEPRECATED . W581, W583
 \best@size x438, x462, x468, x474
 \bf@def@ult A397
 \bfdef@ult A200, A271, A272, A273,
 A314, A315, A316, A401, A442, 579
 \bfdefault@previous A267,
 A270, A310, A313, B106, B116, 1140
 \bfseries@.. 574
 \bfseries@previous 1140
 \bfseries@rm
 ... A105, A128, A207, A210, A231,
 A258, A271, A314, A319, A355, 574
 \bfseries@rm@kernel
 ... A108, A131, A207, A234, 577
 \bfseries@sf
 ... A106, A128, A211, A214, A232,
 A259, A272, A315, A320, A356, 571
 \bfseries@sf@kernel
 ... A109, A132, A211, A235
 \bfseries@tt
 ... A107, A128, A215, A218, A233,
 A260, A273, A316, A321, A357, 577
 \bfseries@tt@kernel
 ... A110, A133, A215, A236
 \bm@b K37
 \bm@c K37
 \bm@l K37
 \bm@r K37
 \bm@s K37
 \bm@t K37
 \botmark 853
 \bx@ 1041
 \bx@A Y30, Y57
 \bx@AA Y40
 \bx@B Y30, Y57
 \bx@BB Y40
 \bx@C Y30, Y57
 \bx@CC Y40
 \bx@D Y30, Y57
 \bx@DD Y40
 \bx@E Y30, Y57
 \bx@EE Y40
 \bx@F Y31, Y58
 \bx@FF Y41
 \bx@G Y31, Y58
 \bx@GG Y41
 \bx@H Y31, Y58
 \bx@HH Y41
 \bx@I Y31, Y58
 \bx@II Y41
 \bx@J Y31, Y58
 \bx@JJ Y41
 \bx@K Y32, Y59
 \bx@KK Y42
 \bx@L Y32, Y59

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

\bx@LL Y42
 \bx@M Y32, Y59
 \bx@MM Y42
 \bx@N Y32, Y59
 \bx@NN Y42
 \bx@O Y33, Y60
 \bx@OO Y43
 \bx@P Y33, Y60
 \bx@PP Y43
 \bx@Q Y33, Y60
 \bx@QQ Y43
 \bx@R Y33, Y60
 \bx@RR Y43
 \bx@S Y38
 \bx@SS Y44
 \bx@T Y38
 \bx@TT Y44
 \bx@U Y38
 \bx@UU Y44
 \bx@V Y38
 \bx@VV Y44
 \bx@W Y39
 \bx@WW Y45
 \bx@X Y39
 \bx@XX Y45
 \bx@Y Y39
 \bx@YY Y45
 \bx@Z Y39
 \bx@ZZ Y25, Y45, Y55
 \c@bottomnumber . . P269, P274, Y2330
 \c@dbltopnumber
 P268, P283, P297, Y2337
 \c@enumi J227
 \c@enumii J227
 \c@enumiv J227, 1082
 \c@equation I349, I383, I543
 \c@errorcontextlines I212, 1086
 \c@footnote O12, P397, P525
 \c@localmathalphabets
 z137, z150, z282, z315, 543
 \c@mpfootnote K357, P399
 \c@ncel B461, B462
 \c@page
 F3, F6, F7, S305, S355, Y140, Y1822
 \c@secnumdepth . . O56, O71, O81, O140
 \c@tocdepth O140, O205, O228
 \c@topnumber P267, P271, Y2326
 \c@totalnumber P270, P276, Y2333
 \c@totalpages X348, X441
 \calculate@math@sizes v645, x219
 \catcodetable@atletter d93, d246
 \catcodetable@initex d93, d243
 \catcodetable@latex d93, d245
 \catcodetable@string d93, d244
 \cdp@elt v96, v116,
 v127, v128, v149, v152, v154, z352,
 z468, z543, z617, z698, aa472, aa473
 \cdp@list
 v98, v114, v128, v156, v157,
 z370, z470, z545, z619, z700, aa473
 \cf@encoding s34, s41,
 s44, s51, s154, v256, v266, v276, v326
 \ch@ck U1153,
 U1172, U1284, U1305, U1397,
 b206, b207, b208, b209, b210,
 b236, b248, b249, b250, b251, b279,
 b281, b293, b294, b295, b296, b302
 \char@if@alph f806
 \chardef@text@cmd s3
 \check@command f187, f189
 \check@icl
 D9, D44, D49, D55, D63, D70, D72
 \check@icr D9,
 D44, D50, D56, D64, D73, D78, 1119
 \check@mathfonts q5,
 s302, s328, s360, s1254, v359, v363,
 v370, v372, x250, z320, z403, E672, 549
 \check@nocorr 1102
 \check@nocorre D46, 1121
 \check@orange x379, x380
 \check@single x378, x400
 \cl@ckpt r440, t35
 \cl@page F4
 \col@number
 Y97, Y150, Y210, Y222, 1095
 \color@begingroup v668,
 v728, I104, I134, K29, K89, K176,
 K345, K391, K408, K425, L47,
 L51, P475, P493, P511, Y495, 1104
 \color@endbox K89, P253, P348,
 P366, Y226, Y635, Y645, Y694, Y704
 \color@endgroup v673, v734,
 I104, I134, K29, K89, K134, K155,
 K178, K373, K395, K412, K428,
 L49, P479, P497, P514, Y499, 1092
 \color@hbox
 K89, Y632, Y642, Y691, Y701, 1106
 \color@setgroup K89, K134, K153, 1097
 \color@vbox K89, P96,
 P165, P339, P357, P381, Y217, 1107
 \conditionally@traceoff
 l289, g221, g233, 338
 \conditionally@traceon l289, g257
 \contionally@traceon 338
 \copy@kernel@robust@command . . f588, f675
 \count@ c14, c15, c16, c17, c18, c20,
 v702, v708, v710, x22, x302, x304,
 x326, x327, z435, z437, z441, z800,

File Key: a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

z801, z802, z848, z849, z850, z909,
 z910, z911, z957, z958, z959, z997,
 z998, z999, z1005, z1006, z1007,
 z1051, z1052, z1053, z1059, z1060,
 z1061, z1120, z1121, z1122, z1128,
 z1129, z1130, a182, a183, a184,
 a189, D115, D118, M730, M731,
 M732, M735, M736, M739, M743,
 M786, M787, M788, M791, M792,
 M795, M799, f169, f173, U1196,
 U1198, U1199, U1200, U1329,
 U1331, U1332, U1333, U1418,
 U1420, U1421, U1422, f287, f312,
 aa239, aa240, aa247, aa249, aa539,
 aa540, aa547, aa549, b41, b191,
 b192, b197, b199, b205, b206, b207,
 b208, b209, b210, b211, a69, b505, b506
 \backslash counterwithin@ s t113, t114, t132
 \backslash counterwithin@ x t113, t116, t133
 \backslash counterwithout@ s t102, t103, t129
 \backslash counterwithout@ x t102, t105, t130
 \backslash curr@fontshape . s180, v88, v388,
 v396, v400, v402, v544, v550, v553,
 v562, v569, v571, v579, v585, v588,
 v596, v603, v605, w454, x92, x100,
 x142, x169, x477, x497, x529, x545,
 x560, z374, z379, A556, A565, 1080
 \backslash curr@math@size
 v377, x256, x262, x267, x284
 \backslash declare@command@copy 1139
 \backslash declare@commandcopy
 f476, f480, f485, f488, f507, 98
 \backslash declare@commandcopy@do
 f488, f534, f535
 \backslash declare@commandcopy@let
 f495, f499, f509, f610, f611, 99
 \backslash declare@environmentcopy
 f517, f521, f526, f529
 \backslash declare@file@substitution
 W247, W568, X517, 937
 \backslash declare@robustcommand f233
 \backslash DeclareEncodingSubset@aux
 E41, E43, E59
 \backslash DeclareFontEncoding@ v122,
 v124, v139, aa383, aa403, aa469, 1133
 \backslash DeclareFontEncoding@saved
 aa383, aa403, aa485
 \backslash DeclareFontShape@ v21, v22
 \backslash DeclareRobustCommand 299
 \backslash DeclareSymbolFont@m@dropped ...
 z460, z465, z508, z509
 \backslash DeclareSymbolFontAlphabet@ ...
 z1175, z1178
 \backslash DeclareUnicodeAccent@
 s1046, s1048, s1052
 \backslash def 316
 \backslash default@ds U444,
 U473, U537, U556, U1023, U1025, 1084
 \backslash default@family v129, v161, v512,
 v526, v529, v554, v589, aa474, 1079
 \backslash default@M
 v136, v176, v179, v183, aa481, 1084
 \backslash default@mextra y10, y89
 \backslash default@series v129, v162,
 v513, v527, v530, v551, v586, aa474
 \backslash default@shape v130, v163,
 v514, v528, v531, v549, v584, aa475
 \backslash default@T v170, v173, v183, v272, 1084
 \backslash define@mathalphabet y18, y131, 1079
 \backslash define@mathgroup y19, y135, 1079
 \backslash define@newfont v380, y389
 \backslash delayed@f@adjustment v290,
 w395, w396, w397, w399, w400,
 w411, w616, w617, w619, w620,
 x122, x126, x134, x138, A662, 468
 \backslash delayed@merge@font@series
 w396, w460, w475, w514, x133, x136
 \backslash delayed@merge@font@shape
 w617, w666, w721, x132, x135
 \backslash development@branch@name
 c11, c39, c53, c54, c55, c62, c63, c64
 \backslash dimen@ l28, l29,
 p350, p355, p384, p389, s429, s430,
 s432, s433, s796, s797, v214, v216,
 v222, v235, v238, v242, v648, v649,
 v650, v654, x451, x452, x453, x454,
 x458, E102, E104, E931, E933,
 I72, I73, I199, I200, I201, I202,
 K490, K493, L176, L177, Y512,
 Y514, Y535, Y537, b41, b502,
 b503, b539, b540, b542, b544, 1118
 \backslash dimen@i b41
 \backslash dimen@ii v218, v223, b41
 \backslash disable@package@load W478, X475, 937
 \backslash display I204, I208, I209
 \backslash do@add@percent@to@temptokena ...
 H138, H144, H162
 \backslash do@emfont@update A557, A561, A581
 \backslash do@noligs H584, H589
 \backslash do@subst@correction v84, x482, x537
 \backslash document@default@language
 r51, r52, r121, r122, Y601, aa286
 \backslash document@select@group
 z139, z145, z387, 543
 \backslash dont@add@percent@to@temptokena ...
 H137, H139, H163
 \backslash dorestore@version z114, z119

File Key: a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

\ds@ U475, U1027, 1084
 \dt@pfalse I205
 \dt@ptrue I204
 \e@alloc d15,
 d49, d79, d89, d178, d182, d186,
 d194, d202, d210, aa12, aa47, b51,
 b52, b53, b55, b56, b63, b64, b66,
 b68, b79, b82, b84, b138, b230, 1139
 \e@alloc@attribute@count
 ... d66, d74, d75, d76, d80, d237
 \e@alloc@bytecode@count
 ... d70, d197, d198, d199, d203, d253
 \e@alloc@ccodetable@count
 ... d67, d84, d85, d86, d90, d241
 \e@alloc@chardef . d48, d178, d194,
 d202, d210, aa12, b60, b102, b211, b212
 \e@alloc@intercharclass@top .. aa35
 \e@alloc@luachunk@count
 ... d71, d205, d206, d207, d211, d255
 \e@alloc@luafunction@count
 ... d68, d173, d174,
 d175, d179, d183, d187, d247, d249
 \e@alloc@top d47,
 d80, d179, d183, d187, d195, d203,
 d211, aa12, b55, b63, b102, b188, b259
 \e@alloc@whatsit@count
 ... d69, d189, d190, d191, d195, d251
 \e@ch@ck d51, d55, b142, b152
 \e@insert@top . b257, b259, b276, b291
 \e@mathgroup@top z56,
 z149, z150, z187, z217, b79, b124, 1129
 \em@currfont A556, A567, 590
 \em@force A565, A570, A573, A583, 590
 \emfontdeclare@clist A546, A548,
 A552, A557, A562, A568, A579, 589
 \empty@sfcnt
 ... x490, x491, x492, x506, x511, x563
 \ENC@cmd 1099
 \enc@update v257,
 v259, v275, v278, x147, x173, 1099
 \end@dblfloat P205
 \end@float P189, P227, P243, P383, 830
 \endlinechar 317
 \enlargethispage 857
 \err@rel@i y12, y99, y132, y136
 \error@fontshape v507,
 v522, v547, v582, x107, x527, z373
 \escapechar 310
 \et@xmaxfam d22, d26, d30, d38
 \et@xmaxregs
 d29, d31, d32, d33, d34, d35, d36, d37
 \every@math@size v78, x235, x247, 1106
 \every@size 1106

\execute@size@function
 ... x362, x390, x404, x421
 \expand@font@defaults A37,
 A148, A256, A279, A309, A330,
 A354, A365, A396, A397, A436,
 A438, A470, A472, A501, E26, E83, 218
 expand@font@defaults A420
 \expandafter 952
 \external@font
 ... x84, x87, x98, x102, x104, x391,
 x405, x467, x501, x569, x571, x573
 \extra@def y9, y84, 1079
 \extract@alph@from@version v622,
 v628, z162, z193, z223, z255, 1082
 \extract@default@composite
 ... s1062, s1069
 \extract@default@composite@a
 ... s1071, s1075
 \extract@default@composite@b
 ... s1073, s1077
 \extract@font v403, x81
 \extract@fontinfo x358, x365
 \extract@rangefontinfo
 ... x375, x382, x401, x434
 \extract@sizefn x350, x372
 \f@... 1099
 \f@baselineskip s870, s1220,
 v317, v324, v533, x121, x167,
 x182, x186, x201, x215, x226, x240
 \f@depth P291, Y347
 \f@encoding s178,
 v251, v270, v273, v274, v276, v326,
 v383, v388, v407, v409, v411, v416,
 v418, v449, v511, v543, v578, w439,
 w443, d260, d275, w652, w656,
 d283, x91, x128, x307, x517, z358, 1099
 \f@family v279, v287,
 v299, v309, v320, v384, v388, v407,
 v409, v411, v416, v418, v450, v529,
 v554, v589, w439, w443, w652,
 w656, x91, x128, z358, z389, A151,
 A178, A179, A258, A259, A260,
 A281, A282, A283, A319, A320,
 A321, A339, A340, A341, A355,
 A356, A357, A366, A367, A368,
 A504, A517, A659, A677, A693,
 E23, E27, E29, E31, E63, E66, E80,
 E84, E86, E88, E93, E100, E897,
 E900, E914, E923, E929, E1144, 1079
 \f@linespread
 ... v320, x120, x166, x183, x184,
 x187, x195, x198, x209, x212, 1099
 \f@series ... q14, v279, v310, v321,
 v385, v388, v530, v551, v586, w400,

File Key: a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

w410, w419, w429, w463, w482,
 w483, w652, w656, x125, x128,
 x131, z390, A144, A160, A162,
 A166, A167, A168, A189, A195,
 A198, A200, A222, A520, A524,
 A660, A678, A694, A739, E7, E613, 503
 \f@series@saved x125, x131
 \f@shape v279,
 v289, v301, v311, v322, v386, v388,
 v531, v549, v584, w439, w443,
 w620, w627, w633, w643, w650,
 w654, w657, w660, w669, w676,
 w678, w713, x124, x128, x130,
 z391, A661, A679, A695, A740, 595
 \f@shape@saved x124, x130
 \f@size s180, s869, s1219,
 v88, v317, v323, v387, v532, v571,
 v605, v647, v648, v651, v652, x121,
 x142, x167, x169, x180, x200, x215,
 x218, x221, x226, x233, x240, x252,
 x255, x261, x267, x284, x285, x288,
 x293, x359, x366, x385, x387, x402,
 x453, x455, x457, x473, x474, x479,
 x493, x505, x510, x522, x530, x535,
 x561, x575, A556, A565, E102, E931
 \f@user@size .. x473, x478, x522, x535
 \f@warn@break 1101
 \famdef@ult A444
 \filec@ntents U1103, U1106,
 U1109, U1120, U1145, U1238,
 U1241, U1244, U1255, U1276,
 U1369, U1391, U1479, U1694, 1111
 \filec@ntents@checkdir
 U1126, U1128,
 U1146, U1261, U1263, U1277, U1376
 \filec@ntents@force
 U1122, U1257, U1372
 \filec@ntents@noheader
 U1124, U1259, U1374
 \filec@ntents@nosearch
 U1125, U1260, U1375
 \filec@ntents@nowarn U1130
 \filec@ntents@opt .. U1106, U1109,
 U1111, U1241, U1244, U1246, U1371
 \filec@ntents@OPTION 908
 \filec@ntents@overwrite
 U1123, U1258, U1373
 \filec@ntents@warning
 U1131, U1133, U1178
 \filec@ntents@where U1127, U1129,
 U1158, U1262, U1264, U1289, U1377
 \filename@area r642,
 r657, r667, r699, r700, r704, r729,
 r732, r749, r765, r767, a249, a255,
 a262, a268, U856, a275, a281, a288
 \filename@base r642, r657,
 r667, r699, r701, r704, r729, r732,
 r756, r765, U857, a298, a305, a318
 \filename@dot a316, a322
 \filename@dots a300, a302, a307
 \filename@ext
 r643, r658, r668, r695, r696, r699,
 r702, r704, r725, r726, r729, r732,
 r757, U858, a297, a304, a314, a316, 943
 \filename@parse a113, r640,
 r655, r665, r694, r724, r754, a245, U855
 \filename@path .. a250, a251, a256,
 a263, a264, a269, a276, a277, a282
 \filename@simple a253, a266,
 a279, a289, a293, a295, a310, a312
 \finish@module@release c89, c93
 \finph@nt
 I104, I106, I110, I111, I119, I120
 \finsm@sh I134,
 I141, I147, I153, I154, I158, I159, 696
 \fix@penalty D101
 \fixed@sfcnt x565, x566, x567
 \fl@ShowFloat .. Y1908, Y1914, Y1929
 \fl@trace . Y242, Y269, Y325, Y353,
 Y360, Y381, Y428, Y476, Y529,
 Y544, Y545, Y546, Y547, Y558,
 Y559, Y560, Y561, Y562, Y572,
 Y785, Y804, Y823, Y841, Y843,
 Y982, Y986, Y998, Y999, Y1000,
 Y1001, Y1007, Y1010, Y1018,
 Y1022, Y1033, Y1038, Y1043,
 Y1044, Y1045, Y1046, Y1053,
 Y1056, Y1064, Y1075, Y1081,
 Y1086, Y1091, Y1097, Y1098,
 Y1103, Y1108, Y1109, Y1110,
 Y1118, Y1122, Y1127, Y1131,
 Y1136, Y1147, Y1148, Y1150,
 Y1168, Y1177, Y1183, Y1192,
 Y1195, Y1201, Y1211, Y1215,
 Y1225, Y1231, Y1237, Y1243,
 Y1250, Y1252, Y1258, Y1263,
 Y1265, Y1267, Y1275, Y1280,
 Y1286, Y1291, Y1297, Y1311,
 Y1312, Y1315, Y1336, Y1345,
 Y1351, Y1360, Y1363, Y1370,
 Y1380, Y1384, Y1396, Y1402,
 Y1407, Y1412, Y1416, Y1420,
 Y1421, Y1428, Y1433, Y1437,
 Y1444, Y1453, Y1457, Y1461,
 Y1462, Y1466, Y1467, Y1477,
 Y1483, Y1489, Y1495, Y1499,
 Y1505, Y1507, Y1515, Y1520,

File Key: a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisenc.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

\Y1525, Y1533, Y1542, Y1547,
Y1552, Y1554, Y1559, Y1561,
Y1572, Y1578, Y1588, Y1594,
Y1598, Y1599, Y1604, Y1605,
Y1611, Y1614, Y1615, Y1616,
Y1623, Y1624, Y1625, Y1633,
Y1638, Y1650, Y1651, Y1658,
Y1661, Y1669, Y1673, Y1677,
Y1678, Y1682, Y1683, Y1693,
Y1699, Y1709, Y1715, Y1719,
Y1720, Y1726, Y1727, Y1734,
Y1737, Y1738, Y1739, Y1747,
Y1748, Y1749, Y1758, Y1763,
Y1776, Y1778, Y1785, Y1788,
Y1797, Y1801, Y1805, Y1806,
Y1810, Y1811, Y1863, Y1868,
Y1874, Y1884, Y1891, Y1905,
Y1906, Y1910, Y1921, Y1933,
Y2031, Y2044, Y2045, Y2049,
Y2052, Y2054, Y2057, Y2060,
Y2062, Y2103, Y2110, Y2115,
Y2121, Y2126, Y2130, Y2136,
Y2144, Y2146, Y2153, Y2158,
Y2163, Y2165, Y2171, Y2173,
Y2180, Y2209, Y2211, Y2226,
Y2228, Y2242, Y2267, Y2271,
Y2276, Y2288, Y2305, Y2310, Y2318
\f1@tracemessage
..... Y1905, Y1922, Y1931, Y1933
\f1@traceval Y1915, Y1916,
Y1917, Y1918, Y1921, Y1930, Y1933
\float@count
..... b51, b52, b53, b62, b188,
b205, b211, b213, b214, b222, b230
\fmtversion@topatch aa660,
aa662, aa674, aa675, aa687, aa695
\font@info x99, x365, x434, x439
\font@name s179,
s182, v86, v194, v196, v379, v394,
v570, v604, x84, x88, x90, x105,
x141, x144, x154, x168, x171, x330,
x331, x332, x333, x334, x339, 1079
\font@submax
.... x441, x470, x471, H49, H51,
H89, H91, aa299, aa301, aa310, 1094
\fps@dbl P34
\freeze@math@version z154, z273
\frozen@everydisplay
.... v349, v353, v368, v370, 547
\frozen@everymath v349, v361, v372, 547
\g@addto@macro i61, w731,
z403, U133, U399, U1035, U1051,
U1052, X352, X358, X408, f918, 226
\G@refundefined 1116
\G@refundefinedfalse G5, 665
\G@refundefinedtrue G3,
G16, G33, G50, R41, R68, R85, 1116
\gen@sfcnt x502, x503, x504
\genb@sfcnt x507, x508, x509, 1115
\genb@x x510, x512
\genb@y x512
\get@cdp z571, z579, z612
\get@external@font x83, x96, x536
\getanddefine@fonts
.... v617, v635, x320, z59, z87, z132,
z159, z190, z220, z251, z294, z333,
z438, z522, z576, z578, z595, z718,
z719, z751, z752, z1194, z1195, 546
\glb@currsize r41, r111, r169,
v346, x217, x252, x256, x262, x285
\glb@settings . v347, x217, x264, x295
\gobble@finish@module@release ...
..... c109, c111, c170
\gobble@font@spec 1113
\group@elt
.... z35, z436, z483, z484, z515, z519, z1226
\group@list
.... z440, z490, z513, z518, z519, z568,
z794, z842, z903, z988, z991, z1042,
z1045, z1111, z1114, z1181, z1232, 1122
\h@false 181
\h@true 182, 183
\hb@xt@ b549, s425, I210, I376, I455,
I470, I482, I509, I540, K44, K65,
K83, K205, K498, K502, K503,
K504, L37, M31, M43, M62, M74,
M105, M117, M265, M299, M302,
M305, M307, M314, M373, M452,
M588, M598, M741, M797, O218,
O241, O248, f29, Y634, Y644,
Y693, Y703, Y1847, Y2261, Y2262,
Y2266, Y2293, Y2294, Y2300, 1110
\hexnumber@
.... z815, z823, z858, z866, z887, z897,
z923, z931, z939, z948, z951, z960,
z961, z1000, z1008, z1054, z1062,
z1081, z1082, z1092, z1093, z1098,
z1124, z1132, z1137, z1139, A645
\hgl@ b504, b505
\hmode@bgroup s67, s75, s327,
s356, s390, s396, s427, s438, s445,
s476, s483, s486, s488, s496, s512,
s730, s760, s766, s798, s805, s833,
s836, s893, s1253, D7, E625, E632, 1124
\hmode@start@before@group
.... s68, s151, s153, s159, s184
\if@afterindent O124, O131
\if@compatibility U2, U613

File Key: a=ltdirname.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\if@endpe H221, H228, H280, H295, [J138](#)
 \if@eqnsw [I357](#), [I423](#)
 \if@fcollmade
 .. [Y97](#), [Y266](#), [Y396](#), [Y405](#), [Y443](#),
 [Y453](#), [Y783](#), [Y803](#), [Y821](#), [Y850](#),
 [Y930](#), [Y2208](#), [Y2225](#), [Y2275](#), [Y2315](#)
 \if@filesw [r5](#), [r36](#),
 [r106](#), [r164](#), [r298](#), [r310](#), [r331](#), [r360](#),
 [r372](#), [r387](#), [r407](#), [r419](#), [r426](#), [r438](#),
 [H21](#), [H55](#), [H81](#), [H95](#), [O153](#), [R4](#), [R8](#),
 [R39](#), [R48](#), [R56](#), [R67](#), [R84](#), [U1156](#),
 [U1173](#), [U1287](#), [U1306](#), [X361](#), [1105](#)
 \if@firstamp [L251](#)
 \if@firstcolumn [Y97](#), [Y248](#), [Y281](#),
 [Y398](#), [Y446](#), [Y1819](#), [Y2239](#), [Y2284](#), [854](#)
 \if@font@series@context
 [A509](#), [A528](#), [A536](#), [587](#)
 \if@forced@series .. [w401](#), [A143](#), [1136](#)
 \if@ignore [z301](#),
 [z340](#), [H4](#), [H223](#), [H230](#), [H281](#), [H296](#)
 \if@in@minipage@env [K326](#), [K348](#)
 \if@includeinrelease
 [c73](#), [c76](#), [c132](#), [f872](#)
 \if@inlabel
 [J28](#), [J65](#), [J102](#), [J160](#), [J183](#), [Y163](#), [Y190](#)
 \if@inmath [1118](#)
 \if@insert
 [Y97](#), [Y1061](#), [Y1173](#), [Y1207](#), [Y1341](#),
 [Y1376](#), [Y1450](#), [Y1539](#), [Y1666](#), [Y1794](#)
 \if@minipage
 [n26](#), [p240](#), [p258](#), [p277](#), [p312](#),
 [H425](#), [H447](#), [J149](#), [K323](#), [L79](#), [P20](#)
 \if@mparswitch [Y97](#), [Y1821](#)
 \if@multiplelabels [G93](#)
 \if@mypkg@draft [925](#)
 \if@negarg .. [M157](#), [M198](#), [M212](#), [M273](#)
 \if@newlist [H467](#), [J29](#), [J33](#), [J69](#), [J78](#),
 [J106](#), [J166](#), [Y603](#), [Y648](#), [Y661](#), [Y707](#)
 \if@nmbrrlist [J33](#), [J201](#)
 \if@no@font@opt [y16](#), [y110](#), [y129](#)
 \if@nobreak [p121](#), [p148](#),
 [p279](#), [p314](#), [r203](#), [r215](#), [J167](#), [J192](#),
 [K286](#), [K307](#), [O47](#), [O128](#), [P180](#),
 [P373](#), [S153](#), [T33](#), [T44](#), [T58](#), [T66](#),
 [Y167](#), [Y194](#), [Y337](#), [Y1152](#), [Y1318](#), [1120](#)
 \if@noitemarg [J32](#), [J199](#)
 \if@noparitem [J30](#), [J157](#)
 \if@noparlist [J31](#), [J114](#)
 \if@noskipsec
 [p148](#), [J58](#), [K287](#), [K308](#),
 [O38](#), [O40](#), [O97](#), [P374](#), [Y157](#), [Y184](#)
 \if@ovb [M427](#),
 [M495](#), [M526](#), [M551](#), [M562](#), [M575](#)
 \if@ovhline [M459](#), [M590](#)

 \if@ovl [M427](#),
 [M493](#), [M522](#), [M547](#), [M592](#), [M601](#)
 \if@ovr [M427](#),
 [M492](#), [M521](#), [M546](#), [M589](#), [M599](#)
 \if@ovt [M427](#),
 [M494](#), [M525](#), [M550](#), [M567](#), [M579](#)
 \if@ovvline [M459](#), [M565](#)
 \if@partsw [r5](#), [r302](#), [r364](#), [r411](#)
 \if@pboxsw [K278](#), [K432](#)
 \if@reversemargin [Y103](#), [Y1824](#)
 \if@reversemarginpar [Y97](#)
 \if@rjfield [L19](#), [L33](#)
 \if@skipping@module .. [c137](#), [c149](#), [c152](#)
 \if@specialpage ... [Y97](#), [Y610](#), [Y668](#)
 \if@tempswa [j9](#), [r308](#),
 [r370](#), [r417](#), [s1542](#), [v99](#), [v712](#), [z471](#),
 [z546](#), [z620](#), [z701](#), [z1225](#), [H57](#), [H97](#),
 [H432](#), [H453](#), [R95](#), [U1185](#), [U1318](#),
 [U1407](#), [W293](#), [W294](#), [W316](#), [W317](#),
 [W337](#), [W338](#), [Y994](#), [Y1030](#), [Y1630](#),
 [Y1755](#), [b270](#), [a81](#), [a82](#), [a83](#), [1127](#)
 \if@test [Y12](#),
 [Y13](#), [Y891](#), [Y910](#), [Y950](#), [Y972](#),
 [Y1036](#), [Y1120](#), [Y1129](#), [Y1278](#),
 [Y1289](#), [Y1431](#), [Y1518](#), [Y1636](#), [Y1761](#)
 \if@twocolumn [r26](#), [r96](#),
 [r153](#), [P32](#), [P210](#), [P235](#), [Y97](#), [Y141](#),
 [Y269](#), [Y280](#), [Y397](#), [Y445](#), [Y469](#),
 [Y785](#), [Y841](#), [Y1818](#), [Y2210](#), [Y2227](#)
 \if@twoside ... [Y97](#), [Y140](#), [Y613](#), [Y671](#)
 \ifdt@p [I203](#), [I205](#)
 \IfFileExists@ [r471](#), [r498](#)
 \IfFileExists@@ [r498](#)
 \ifG@refundefined [G3](#), [G4](#), [G5](#)
 \ifh@ [I76](#), [I114](#), [I123](#)
 \ifin@ [s1558](#), [s1561](#), [y50](#),
 [y52](#), [z1](#), [z22](#), [z425](#), [z567](#), [z569](#), [z630](#),
 [z643](#), [z713](#), [z715](#), [z743](#), [z795](#), [z809](#),
 [z843](#), [z855](#), [z904](#), [z920](#), [z989](#), [z992](#),
 [z1013](#), [z1043](#), [z1046](#), [z1109](#), [z1112](#),
 [z1115](#), [z1182](#), [z1184](#), [z1213](#), [A210](#),
 [A214](#), [A218](#), [U236](#), [U254](#), [U485](#), [U519](#)
 \ifmath@fonts [v204](#), [x222](#)
 \ifmaybe@ic [D82](#), [D91](#), [1127](#)
 \ifnot@nil [x343](#), [x360](#), [x381](#)
 \ifrestore@version [1118](#)
 \iftc@forced [E871](#), [E882](#), [E1150](#)
 \ifv@ [I75](#), [I113](#), [I122](#)
 \ifx [317](#)
 \in@ [s1556](#), [s1559](#), [y49](#), [y51](#), [z1](#),
 [z21](#), [z424](#), [z566](#), [z568](#), [z626](#), [z639](#),
 [z712](#), [z714](#), [z741](#), [z793](#), [z804](#), [z841](#),
 [z852](#), [z902](#), [z916](#), [z987](#), [z990](#), [z1010](#),
 [z1041](#), [z1044](#), [z1106](#), [z1110](#), [z1113](#)

File Key: a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltruatex.dtx, e=ltxpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntr1.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspac.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

z1180, z1183, z1211, A208, A212,
 A216, U235, U252, U482, U518, 500
 $\backslash\text{in@}$ w489, w490, w492, w493, z5, z6, z7, z9
 $\backslash\text{in@false}$ z10
 $\backslash\text{in@true}$ z12
 $\backslash\text{init@restore@glb@settings}$ x265, x268, x270
 $\backslash\text{init@restore@version}$ z62, z91, z108, z123, z124, 1118
 $\backslash\text{init@series@setup}$ w732, w737, A120, A206, A246
 $\backslash\text{input@path}$ a112, r484, r538, r560, r587, r604,
 a134, a136, a142, a144, a150, a152,
 a157, a159, a169, a236, W276, 949
 $\backslash\text{insc@unt}$ Y61, 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
 $\backslash\text{install@mathalphabet}$ v612, v629, v636, z444,
 z447, z573, z574, z671, z723, z726,
 z733, z748, z749, z756, z1196, z1198
 $\backslash\text{is@range}$ x376, x377
 $\backslash\text{kern}$ 858
 $\backslash\text{kernel@ifnextchar}$ c90, f81, U379, f100, f150, f783, f798
 $\backslash\text{kernel@make@fragile}$ p25, p26, p27, p28,
 p29, s172, s173, H401, H402, H403,
 I90, I91, I92, I93, I179, I180, I181,
 L160, L161, L162, M823, M824,
 M825, M826, M827, M828, M829,
 M830, M831, M832, M833, M834,
 O23, O24, O25, O26, O27, T72,
 T73, f387, f899, f900, f901, f902,
 f903, f904, f905, f906, f907, f908,
 f909, f910, f911, f912, f913, f914, 30
 $\backslash\text{l@ngrel@x}$ f74, f75, f76, f120, f167
 $\backslash\text{l@nohyphenation}$ H429, H569, aa283, 1132
 $\backslash\text{last@fontshape}$ v545, v563, v580, v597
 $\backslash\text{latexrelease@postltcmd}$ g2899
 $\backslash\text{latexrelease@postltexpl}$ e132
 $\backslash\text{leavevmode@ifvmode}$ p463, p464, p472, B624, B626,
 B628, B630, I115, I154, I219, I239, I240
 $\backslash\text{load@onefile@withoptions}$ U860, U902, U1014, 900
 $\backslash\text{load@onefilewithoptions}$ U810, U964, U1494, 1151
 $\backslash\text{lower@bound}$ x386, x387, x398
 $\backslash\text{lst@vskip}$ 353
 $\backslash\text{ltx@sh@ft}$ s390,
 s397, s476, s484, s760, s767, b541, 1127
 $\backslash\text{m@ne}$ b39, 322
 $\backslash\text{m@th}$ q13, B337, B459, B461,
 B462, B465, B506, B530, B533,
 B537, B540, B547, B550, B557,
 B560, B642, I68, I71, I106, I140,
 I147, I167, I169, I185, I204, I367,
 I470, I482, I509, I520, K278, K458,
 L181, O214, O237, P400, P409,
 P416, P437, P444, b521, b533, 1121
 $\backslash\text{makeph@nt}$ I101, I103
 $\backslash\text{makesm@sh}$ I131, I133
 $\backslash\text{mandatory@arg}$ x414, x501,
 x505, x510, x517, x519, x524, x526,
 x531, x533, x546, x562, x569, x571
 $\backslash\text{math@bgroup}$ v643, x306, x312,
 z53, z81, z146, z175, z184, z206,
 z214, z245, D130, D131, D138, 544
 $\backslash\text{math@egroup}$ v643,
 x310, x311, D131, D132, D139, 1080
 $\backslash\text{math@famname}$ 1079
 $\backslash\text{math@fonts}$ v613, v618, x232,
 x336, z60, z89, z160, z191, z221, z253
 $\backslash\text{math@fontsfalse}$
 . q7, s302, s329, s360, s1255, v77,
 v206, v216, v239, E103, E673, E932
 $\backslash\text{math@fontstrue}$ v204, v655
 $\backslash\text{math@group}$ 1079
 $\backslash\text{math@version}$ v7, v336, v617, v621,
 v623, v624, v626, x230, z56, z59,
 z64, z65, z69, z84, z88, z93, z94,
 z98, z111, z112, z113, z126, z127,
 z128, z149, z151, z153, z154, z159,
 z163, z165, z167, z171, z187, z190,
 z194, z196, z198, z202, z217, z220,
 z224, z226, z228, z232, z248, z252,
 z256, z258, z260, z264, A617, 1079
 $\backslash\text{math@version.}$ 1079
 $\backslash\text{mathchar@type}$ z887, z897, z948,
 z951, z960, z976, z1081, z1092, z1155
 $\backslash\text{mathph@nt}$ I99, I105
 $\backslash\text{mathsm@sh}$ I129, I138, I139, I145, I146
 $\backslash\text{maybe@ic}$ D63, D64, D83
 $\backslash\text{maybe@ic@}$ D83
 $\backslash\text{maybe@icfalse}$ D97
 $\backslash\text{maybe@icttrue}$ D87
 $\backslash\text{maybe@load@fontshape}$
 . s71, w476, w515, x127, A164, 410
 $\backslash\text{maybe@update@bfseries@defaults}$ A38, A257, A266, A304
 $\backslash\text{maybe@update@mdseries@defaults}$ A39, A280, A289, A305

File Key: a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdfnns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmdbhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspacel.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfnctcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=ltxref.dtx, H=ltmisenc.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

\mb@b K55, K66, K74, K84
 \mb@l K55,
 K59, K65, K74, K78, K83, M137, M141
 \mb@r K55,
 K59, K65, K74, K78, K83, M137, M141
 \mb@t K56, K63, K75, K82
 \mddef@ult A198, A293, A294, A295,
 A334, A335, A336, A402, A443, 1137
 \mddefault@previous A290,
 A292, A331, A333, B107, B117, 1140
 \mdseries@.. 572
 \mdseries@previous 1140
 \mdseries@rm
 A111, A127, A128, A229,
 A237, A281, A293, A334, A339, A366
 \mdseries@sf A112, A128,
 A238, A282, A294, A335, A340, A367
 \mdseries@tt A113, A128,
 A239, A283, A295, A336, A341, A368
 \meaning 319
 \merge@font@series w408,
 w425, w426, w507, w509, x133, 498
 \merge@font@series@ w428, w433, w510
 \merge@font@series@without@substitution
 w460, w475, w512, x136, 498
 \merge@font@series@without@substitution@
 w460, w513
 \merge@font@shape
 w626, w639, w640, w711, w716, x132
 \merge@font@shape@ . w642, w647, w717
 \merge@font@shape@without@substitution
 w666, w719, x135
 \merge@font@shape@without@substitution@
 w666, w720
 \mv@{version} 546
 \mv@{version}@frozen 543
 \mv@{version}@reset 546
 \n@space B625, B627, B629,
 B631, B636, B637, B638, B639, B642
 \new@command
 f78, f77, f131, f165, f184, f251
 \new@environment f147, f146, f159
 \new@fontshape y2, y4, y22, y24
 \new@mathalphabet ... z624, z645, z656
 \new@mathgroup d27,
 v14, z474, b78, b80, b98, b100, 1112
 \new@mathversion
 z20, z407, z415, z420, z423
 \new@module@skip c138, c150, c152
 \new@moduledate .. c82, c101, c104, c152
 \new@modulename c96, c152
 \new@symbolfont z475, z517
 \newcommand 299
 \NewCommandCopy 306
 \NewDocumentCommand 299
 \newlinechar 317
 \newmathalphabet@ y14
 \newmathalphabet@00 y109
 \newmathalphabet@000 y15, y109
 \nfss@catcodes
 v19, v120, v412, v413,
 v420, v467, B40, B45, B135, Y3, 1108
 \nfss@text s315, s317,
 A648, D5, D122, G17, G34, G51, 1082
 \no@alphabet@error v4, z443, z445,
 z661, z662, z676, z685, z771, z772, 1094
 \no@alphabet@help 1094
 \no@version@warning 1080
 \noaccents@ v658, B129
 \noexpand 319
 \non@alpherr v637, v639, z72, z101,
 z117, z174, z205, z235, z267, z1233
 \not@base A720,
 A724, A725, A726, A727, A728,
 A729, A730, A731, A732, A733, A734
 \not@math@alphabet w527,
 w532, w537, w683, w687, w690,
 w693, w696, w699, w702, w705,
 A5, A8, A11, A14, A17, A20, A23,
 A26, A29, A255, A278, A308, A329,
 A353, A364, A381, A384, A406,
 A411, A416, A449, A454, A459,
 A475, A478, A481, A484, A487, A597
 \now@and@everyjob .. s1011, d215, d221
 \o@align
 s390, s397, s476, s484, s760, s767, b535
 \on@line
 . I8, I15, I214, A144, A147, H184,
 H239, H256, H289, K150, U868,
 U1003, h531, h602, h1428, h1460, 347
 \operator@font B643,
 I3, I4, I5, I6, I7, I8, I9, I10, I11,
 I12, I13, I14, I15, I16, I17, I18, I19,
 I20, I21, I22, I23, I24, I25, I26, I27,
 I28, I29, I30, I31, I32, I33, I34, I37, I40
 \optional@arg
 x415, x494, x496, x568, x571
 \outer@nobreak
 . P181, P251, P255, P346, P364, 1107
 \p@ b311
 \p@... 1134
 \p@enum 722
 \p@equation I364, I518
 \p@renwd 1090
 \p@selectfont x119, 1080
 \par 165
 \par@deathcycles ... J56, J77, J79, J80

File Key: a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphe.dtx, aa=ltfinal.dtx

```

\patch@level .... c1, c39, c44, c46,
    c48, c52, c61, aa663, aa675, aa677, 1130
\patchcmd ..... 299
\ph@nt ..... I81, I82, I83, I97
\pickup@font s181, v195, v378, v572,
    v606, x143, x170, x331, x333, x335
\pictur@ ..... M21
\pkgcls@arg ..... U1505, U1628
\pkgcls@candidate . U1492, U1507,
    U1583, U1587, U1591, U1659, U1662
\pkgcls@debug ..... U1482,
    U1498, U1499, U1500, U1501,
    U1502, U1559, U1560, U1561,
    U1562, U1571, U1576, U1594,
    U1603, U1618, U1652, U1653, U1654
\pkgcls@innerdate ..... U1487, U1532, U1535, U1541, U1680
\pkgcls@mindate ..... U1512, U1521, U1537, U1542, 918
\pkgcls@name ..... U1504, U1547
\pkgcls@parse@date@arg . U1506, U1517
\pkgcls@parse@date@arg@ U1523, U1526
\pkgcls@parse@date@arg@version ..
    U1533, U1554
\pkgcls@releasedate ..... U1492, U1588, U1592, U1663
\pkgcls@rollbackdate@error .. U1584, U1642, U1660
\pkgcls@show@selection ..... U1611, U1616, U1666, U1671
\pkgcls@targetdate U1487, U1519,
    U1527, U1530, U1531, U1535,
    U1543, U1544, U1557, U1565,
    U1580, U1582, U1612, U1623,
    U1625, U1650, U1656, U1658, 918
\pkgcls@targetlabel ..... U1487,
    U1520, U1540, U1555, U1567,
    U1599, U1632, U1670, U1673, 918
\pkgcls@use@this@release . U1568,
    U1585, U1587, U1600, U1610, U1662
\pr@@cs ..... I259, I267
\pr@@ct ..... I262, I268
\pr@mcs ..... I256, I257
\preload@sizes ..... y11, y94, 1079
\prepare@family@series@update ...
    A117, A141, A243, A407,
    A412, A417, A450, A455, A460, 584
\pretocmd ..... 299
\prim@s ..... I253, I255, I267
\prime@s ..... I254
\process@table .. r40, r110, r168, z351
\protectd@edf ..... 1148
\protected ..... 278
\protected@cmd ..... 1112
\protected@edef ..... t192, A546,
    G116, G120, G122, G143, G147,
    G154, G158, G165, K389, K406,
    K423, O60, P472, P490, P508,
    U476, U832, U1075, f222, f265, 418
\protected@file@percent ..... H129, H136,
    H151, H159, H160, O165, O172, 812
\protected@wlog ..... U325, U327, U341, U355
\protected@write ..... r202, r207,
    G105, G134, O181, Q14, Q31, 1117
\protected@xdef ..... O11, P453, P521, P537,
    U317, U336, U389, U729, f265, 1105
\provide@command ..... f179, f178
\ps@empty ..... T10, aa158
\ps@plain ..... T13
\q@curr@file ..... U1103,
    U1147, U1149, U1154, U1180,
    U1278, U1280, U1285, U1313, 1139
\quote@@name ..... r462, r478
\quote@name ..... r462,
    r475, r477, r588, r590, r599, U1278
\r@@t ..... I66
\reenable@package@load ..... W478, X464, 981
\reinstall@nfss@defs ..... w685,
    w726, w730, w732, w736, w737, w740
\relax ..... 319
\rem@pt ..... v329
\remove@angles ..... x347, x370
\remove@nil ..... z36
\remove@star ..... x347, x353
\remove@tlig .. s1001, s1003, s1005,
    s1029, s1034, s1081, s1082, s1083, 1139
\remove@to@nnil v328, x347, x373, x486
\renew@command .. f125, f124, f185, f193
\renew@environment ..... f153, f152
\requested@test@context ..... A503, A520, A524, 587
\reserved@@b ..... 560
\reserved@a ..... k33,
    k37, c13, l234, c19, c34, p410, p413,
    r212, r213, r235, r244, r253, r306,
    r368, r415, a124, r485, r487, r492,
    r494, r506, r509, r512, r515, r516,
    r517, c181, c182, r539, r541, r546,
    r548, r549, r550, r561, r563, r568,
    r570, r585, r591, r595, r602, r608,
    r612, r641, r644, r645, r647, r656,
    r659, r666, r669, r691, r692, r693,
    r697, r705, r722, r723, r727, r733,
    r755, r759, r762, r767, a128, s81,

```

File Key: a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr1.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=lpictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

s82, s86, s89, s97, s107, a129, s110, s119, s138, s143, s995, s999, s1043, s1045, s1046, s1048, s1050, s1058, s1067, v30, v31, v32, v41, v44, v47, v63, v66, v69, v105, v108, v110, v147, v151, v414, v417, v544, v545, v560, v563, v568, v579, v580, v593, v597, v602, v629, v632, v633, v641, w435, w436, w439, w440, w455, w456, w468, w469, w648, w649, w652, w653, w674, w675, x196, x198, x200, x210, x212, x215, x344, x345, x358, x359, y53, y57, z460, z463, z535, z538, z571, z580, z582, z626, z629, z639, z642, z740, z742, z804, z808, z852, z854, z915, z918, z1010, z1012, z1106, z1108, z1210, z1212, z1228, z1230, z1231, z1236, A40, A41, A73, A74, A174, A175, A516, A517, D47, D48, D53, D54, D65, D68, D88, D95, a198, a199, a202, H127, H128, H182, H183, H188, H237, H238, H242, H254, H255, H259, H287, H288, H292, H303, H304, I418, I419, I420, I421, I423, a220, a224, K57, K58, K61, K76, K77, K80, K145, K151, e22, e23, e24, e25, e26, L241, L245, L250, L269, e47, L360, L361, e52, M199, M201, M205, e89, e95, e105, M478, M479, M506, M507, M535, M536, P29, P30, P32, P33, P63, P67, P72, P74, P76, P78, P83, P84, P132, P136, a246, P142, P145, P148, P151, a253, a256, a258, a259, U131, U134, U136, U230, U238, U242, U248, U256, U260, U387, U389, U390, U391, U395, U410, U412, U413, U414, U418, a266, U593, U597, U603, U607, f117, f120, U685, U686, U689, U731, U735, U747, U748, U750, U759, U763, U775, U776, U778, U786, U790, U802, U803, U804, U806, f133, f134, U818, f135, U821, U822, U824, f137, a269, U905, U954, U971, a271, U1012, a272, U1114, U1115, U1117, U1249, U1250, U1252, U1294, U1295, U1297, U1301, f184, f185, f186, f192, f193, f194, f195, f198, U1509, U1514, U1566, U1567, U1598, U1599, U1669, U1670, U1694, U1696, f222, a279, f228, f238, f242, a282, W161, W167, W168, W169, a284, f300, f304, f338, f342, f366, f370, Y37, Y46, Y48, Y50, Y881, Y901, f481, f489, f490, f522, f530, f531, f532, f533, Y2001, Y2003, Y2004, Y2093, Y2095, Y2101, Y2104, f573, f576, aa226, aa243, aa244, aa245, f591, aa252, aa253, aa254, f593, f596, f605, aa491, aa494, aa525, aa531, aa532, aa543, aa544, aa545, aa552, aa553, aa554, aa579, aa580, aa584, aa587, aa589, aa591, aa596, aa599, aa607, aa608, aa609, aa676, aa679, aa680, aa697, f627, f634, f659, f661, a334, a335, a336, f780, f789, b193, 1107
\reserved@b k33, k34, k37, p411, p412, p419, r304, r306, r366, r368, r413, r415, a125, r586, r588, r590, r603, r605, r607, a126, r691, r692, r693, r758, r760, r761, r768, s90, s97, s112, s119, s998, s999, s1044, s1045, s1067, s1076, s1078, v31, v32, v38, v95, v97, v150, v151, v630, v641, w454, w455, w457, y47, y54, y71, y73, z461, z462, z463, z467, z469, z536, z537, z538, z542, z544, z579, z580, z581, z616, z618, z697, z699, z744, z745, z746, z753, z913, z917, z919, A181, A186, A190, A191, A198, A199, D52, D53, D66, D68, D95, D96, L246, L248, L250, e48, e54, P43, P44, P112, P113, U231, U232, U233, U235, U250, U253, U387, U410, f109, f111, f118, U739, U745, U748, U767, U773, U776, U794, U800, U804, U818, f135, U825, f136, U1163, U1164, U1167, U1168, U1203, U1204, U1206, U1233, U1296, U1297, U1300, U1301, U1336, U1337, U1339, U1365, U1425, U1426, U1428, U1455, f239, f240, f242, f301, f302, f304, f339, f340, f342, f367, f368, f370, Y790, Y793, Y807, Y810, Y827, Y830, f574, f576, aa229, aa231, aa235, f592, f596, aa497, aa499, aa503, aa697, f631, f634, f781, f791
\reserved@c a126, r748, a131, v96, v97, v631, v634, y48, y55, y61, y68, z33, z37, z468, z469, z543, z544, z617, z618, z698, z699, z721, z730, z745, z759, z1000, z1017, z1026, z1054, z1065, z1123, z1136,

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=lttoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

z1138, A183, A186, A196, A197, x502
 A200, A201, D67, D69, D76, U819, x507
 U821, U822, U1148, U1153, U1154, x502
 U1172, U1180, U1186, U1208, x507
 U1216, U1279, U1284, U1285, x514
 U1305, U1313, U1319, U1341, x557
 U1348, U1396, U1397, U1398, x557
 U1408, U1430, U1437, U1465, aa233, aa238, aa246, aa491, aa501, x502
 aa522, aa523, aa524, aa526, aa527, x507
 aa528, aa529, aa530, aa538, aa546, x514
 aa699, f786, f789, f791, f794, 1121
 \reserved@d r746, r748, a129, x514
 a132, y61, y68, y70, y74, z1008, x514
 z1017, z1026, z1062, z1065, z1131, x514
 z1136, z1140, A188, A189, A194, x514
 A195, e46, e51, aa700, f779, f788, 1121
 \reserved@e p58, p60, p70, x514
 p72, p101, p108, p116, y39, y45, x514
 y70, y73, y74, z34, z39, aa701, 1114
 \reserved@f p59, x514
 p60, p71, p72, p116, s1543, s1544, x514
 s1545, s1546, s1548, s1555, v190, x514
 v192, v198, v199, x382, x393, x397, x514
 x401, x407, x410, x449, x486, x489, x514
 y27, y38, y45, y71, y73, aa702, 1107
 \reset@font A219, A653, A684, A699, x514
 A714, G17, G34, G51, K385, K403, x514
 K420, P175, P371, P466, P485, x514
 P503, R40, T14, Y620, Y679, 1081
 \restglob@settings x268, x278
 \restore@mathversion
 z107, z110, z125, z133
 \restore@protect f265
 \rlh@ B464, B465
 \rm@def@ult A397
 \rmdef@ult A258, A281, A319, A339, x514
 A355, A366, A398, A439, E27, E84, 579
 \robust@command@act i90, x514
 f422, f423, f425, f493, f550, f687, 96
 \robust@command@act@chk@args ...
 f447, f468, 96
 \robust@command@act@do . f433, f465, 97
 \robust@command@act@end ...
 f430, f431, f443, f446, f466, 97
 \robust@command@act@loop ...
 f427, f433, f463, 97
 \robust@command@act@loop@aux ...
 f433, f464
 \robust@command@chk@safe .. i137,
 f315, f426, f447, f467, f607, f619, 93
 \s@fct@ x426, x490
 \s@fct@alias x552
 \s@fct@fixed x565
 \s@fct@gen x502
 \s@fct@genb x507
 \s@fct@sgen x502
 \s@fct@sgenb x507
 \s@fct@sub x514
 \s@fct@subf x557
 \saved@space@catcode .. aa355, aa424
 \scan@@fontshape y7, y40, y43
 \scan@fontshape y6, y26, y37
 \scantokens 317
 \scriptfont@name x333, x338
 \section 300
 \select@group v614, v633, z48, z387,
 z448, z626, z679, z688, z726, z758, 543
 \series@change@debug A137,
 A144, A147, A158, A161, A165,
 A177, A185, A190, A196, A199, A201
 \series@check@toks . w490, w492, w499
 \series@drop@one@m . w496, w500, w518
 \series@maybe@drop@one@m
 ... v31, w483, w485, w517, z462,
 z537, A168, A187, A193, A401, A402
 \series@maybe@drop@one@m@x
 w486, w488
 \seriesdefault@kernel A220, A773, 596
 \set@mathdelimiter z1063, z1097
 \set@color K88
 \set@curr@file
 ... r226, r235, r262, r270, r452,
 r470, U847, U1146, U1277, W267, 901
 \set@curr@file@aux W273, W279, W280
 \set@curr@file@nosearch . W267, 1147
 \set@display@protect 17,
 l14, l34, l61, f8, f16, U328, f263, 1105
 \set@fontsize
 ... v317, v319, x121, x167, x177, 1097
 \set@mathaccent
 ... z813, z821, z856, z864, z882
 \set@mathchar z937, z947
 \set@mathdelimiter z1014, z1023, z1075
 \set@mathradical z395, z1133
 \set@mathsymbol z921, z929, z950
 \set@simple@size@args
 ... x348, x361, x368, x389, x403
 \set@size@funct@args x351, x353, x411
 \set@size@funct@args@ x411
 \set@target@series
 ... v288, v300, w437, w441,
 w444, w447, w470, w472, w481, w516
 \set@typeset@protect
 ... L197, L235, f263, f282, X84,
 X128, Y607, Y609, Y665, Y667, 1119
 \SetMathAlphabet@ ... z633, z702, z711
 \SetSymbolFont@ z493, z547, z565

File Key: a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

```

\SetSymbolFont@m@dropped .....
..... z535, z540, z560, z561
\sf@def@ult ..... A397
\sf@size .....
.. q6, s302, v224, v243, v653, x328,
x332, E672, P409, P416, P437, P444
\sfdef@ult A259, A282, A320, A340,
A356, A367, A399, A440, E29, E86, 579
\sh@ft ..... b539, 1127
\show@kernel@robust@command .....
..... f613, f676, f704
\show@release@info .....
.. c38, c41, c46, c51, H41, H42, H44, 39
\ShowCommand ..... 306
\sixt@n .....
.. d30, v14, z84, z248, z799, z801,
z847, z849, z908, z910, z956, z958,
z996, z998, z1004, z1006, z1050,
z1052, z1058, z1060, z1119, z1121,
z1127, z1129, M278, M293, M295,
P62, P80, P131, P153, Y1009,
Y1055, Y1194, Y1362, Y1596,
Y1660, Y1717, Y1787, Y2047,
Y2056, Y2112, Y2128, Y2161,
b16, b64, b66, b96, b97, b98, a74
\sixt@nl ..... 322
\size@update .....
..... x146, x172, x185, x204, x206
\sizefn@info .....
..... x352, x354, x362, x390, x404
\skip@ ..... D105,
D108, b41, b501, b503, b504, b506
\sp@ce@skip ..... p84, p443, p444
\sp@n ..... L379
\split@name .....
.. v382, v394, v508, v523, x519, x533
\ssf@size ..... s328, s360,
s1254, v225, v244, v654, x328, x334
\string@makeletter ..... U824,
U856, U857, U858, W169, f806, 1140
\strip@meaning ..... 1090
\strip@prefix ..... a114, v611,
a231, f240, f302, f340, f368, a326, f803
\strip@pt .....
.. v216, v222, v223, v224, v225, v238,
v242, v329, v653, v654, x180, b543
\sub@sfcnt .....
.. x514, x515, x516, x543
\subf@sfcnt ..... x557, x558, x559
\subst@correction ..... v85, v91
\subst@fontshape ..... y8, y80
\subst@size ..... x465
\sw@slant ..... D91, D101
\t@st@ic ..... D90, D94, 1085
\target@meta@family@value .....
..... A150, A175, A182, A184
\target@series@value ..... A149,
A157, A160, A162, A166, A167,
A168, A191, A197, A198, A200, 584
\tc@check@accent .....
.. E109,
E161, E163, E165, E167, E169,
E171, E173, E175, E177, E179,
E181, E183, E185, E187, E189,
E191, E938, E1014, E1016, E1018
\tc@check@symbol .....
.. E109,
E211, E213, E215, E217, E219,
E221, E223, E225, E227, E229,
E231, E233, E235, E237, E239,
E241, E243, E245, E247, E249,
E251, E253, E255, E257, E259,
E261, E263, E265, E267, E269,
E271, E273, E275, E277, E279,
E281, E283, E285, E287, E289,
E291, E293, E295, E297, E299,
E301, E303, E305, E307, E310,
E312, E314, E316, E318, E320,
E322, E324, E326, E328, E330,
E332, E334, E336, E338, E340,
E342, E344, E346, E348, E350,
E354, E356, E358, E360, E362,
E364, E366, E938, E1008, E1010,
E1012, E1020, E1022, E1024,
E1026, E1028, E1030, E1032,
E1034, E1036, E1038, E1040,
E1042, E1044, E1046, E1048,
E1050, E1052, E1054, E1056,
E1058, E1060, E1062, E1064,
E1066, E1068, E1070, E1072,
E1074, E1076, E1078, E1080,
E1082, E1084, E1086, E1088,
E1090, E1092, E1094, E1096,
E1098, E1100, E1102, E1104,
E1106, E1108, E1110, E1112,
E1114, E1116, E1118, E1120, E1122
\tc@error ..... E110, E918, E939
\tc@errorwarn .....
.. E21, E76, E78, E825, E827,
E829, E830, E877, E878, E879, E912
\tc@fake@euro E97, E352, E926, E1007
\tc@forcedfalse ..... E871
\tc@forcedtrue ..... E876
\tc@oldstylesubst ..... E16, E20
\tc@subst ..... E77, E109, E911, E938
\tc@swap@accent ..... E112, E113
\test@font@series@context .....
.. A505, A515, A535, 587
\test@next ..... 1113
\text@command ..... D8, D46, 1102

```

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

```

\textrmfont@name ..... x331, x337
\tf@size ..... v223, v243, v652, x328, x330, 1083
\thr@@ ..... b563, b573, b607, x58, x254, x260, x273, x280, x287, x292, I376, I539, J232, J243, M287, M288, M290, M291, M329, M355, M388, M411, aa84, aa92, b16, 1110
\toks@ ..... c94, c106, c108, p409, p410, p415, c115, c119, c122, c127, v148, v152, v154, v157, v221, v226, z6, z7, z434, z438, z444, z447, z452, z518, z519, z521, z522, z572, z574, z578, z595, z598, z657, z669, z670, z671, z717, z719, z725, z733, z737, z749, z752, z755, z763, z765, z1186, z1188, z1190, z1193, z1195, z1198, z1201, z1233, z1234, U440, U441, U443, U444, Y2247, Y2248, Y2249, Y2250, f920, f921, b41, 500
\topmark ..... 853
\topmark(s) ..... 853
\try@load@font@shape ..... 1122
\try@load@fontshape .. v397, v405, v451, v556, w479, x520, z359, z376, 499
\try@simple@size ..... x356, x481
\try@simples ..... x439, x445, x449
\try@size@orange .... x101, x356, x432
\try@size@substitution ... x103, x436
\tryif@simple ..... x447, x448
\tryis@simple ..... x448
\tt@def@ult ..... A397
\ttdef@ult A260, A283, A321, A341, A357, A368, A400, A441, E31, E88, 579
\tw@ ..... b16, 1102
\two@digits ..... x512, a188, a189, f2, U1099, U1192, U1325, U1414, a89
\type@restoreinfo ..... x202, x207
\undeclare@... ..... 981
\undeclare@file@substitution ... ..... W247, 937
\unexpandable@noexpand ..... 1116
\unexpanded ..... 849
\unqu@tefilef@und ..... W143
\unquote@name .. r456, r462, r479, W350
\unrestored@protected@xdef .... P458, P526, P542, P553, S141, T26, T54, T78, f265
\unskip ..... 857
\update@series@target@value ... ..... A118, A152, A173, A244
\update@uclc@with@cyrillic .... s1505, s1533, s1563, s1571
\upper@bound .. x383, x384, x385, x398
\use@mathgroup ..... v620, v638, v640, x299, z63, z92, z639, z741, z744, z1211, z1235, 1080
\UTF@two@octets@noexpand ..... 1147
\UTFviii@four@octets ..... aa410, aa415, aa421
\UTFviii@four@octets@combine .. aa445
\UTFviii@four@octets@noexpand .. aa451
\UTFviii@four@octets@string .. aa448
\UTFviii@invalid ..... aa349, aa442
\UTFviii@invalid@err ..... aa407, aa412, aa418
\UTFviii@invalid@err@... .. aa407, aa418
\UTFviii@three@octets ..... aa409, aa414, aa420
\UTFviii@three@octets@... aa409, aa420
\UTFviii@three@octets@combine .. aa444
\UTFviii@three@octets@noexpand aa450
\UTFviii@three@octets@string .. aa447
\UTFviii@two@octets ..... aa408, aa413, aa419
\UTFviii@two@octets@... .. aa408, aa419
\UTFviii@two@octets@combine .. aa443
\UTFviii@two@octets@noexpand .. aa449
\UTFviii@two@octets@string .. aa446
\UTFviii@undefined@err ..... aa406, aa411, aa417
\UTFviii@undefined@err@... aa406, aa417
\v@false ..... 182
\v@true ..... 181, 183
\vbox ..... 857
\ver@... ..... 401
\ver@... ..... 401
\ver@{file}.(ext) ..... 900
\verb@balance@group .. H515, H517, H530, H532, H545, H547, H553, H554
\verb@egroup ..... H515, H518, H530, H533, H545, H548, H554, H558
\verb@eol@error .. H555, H567, H577
\verbatim@font ..... H439, H460, H468, H568, H578, 1085
\verbatim@nolig@list .. H583, H589
\verbatim@out ..... 1121
\version@elt ..... z18, z31, z32, z431, z432, z491, z521, z632, z670, z762, z1191
\version@list ..... z16, z21, z32, z424, z432, z496, z527, z566, z637, z682, z712, z767, z1204
\vglo ..... b501, b502
\voidb@x ..... u18, b311, b532
\vspli ..... 857
\vs ..... 857

```

File Key: a=ltdirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

```

\warn@rel0i ..... y5,
    y25, y29, y81, y85, y90, y95, y119, y140
\wrong@fontshape ..... v401, v538, 1079
\x@protect ..... f243,
    f254, f305, f343, f371, f577, f597, 1105
\xe@alloc@ ..... aa56, aa66
\xe@alloc@intercharclass . aa35, 1129
\xe@ch@ck ..... aa57, aa61
\XXX@argdef ..... 1092
\z@ ..... b311, 322
\z@skip ..... b311, 353
\zap@space .. r262, r282, A546, R49,
    U231, U392, U415, U427, U592,
    U708, U729, U747, U758, U775,
    U785, U802, U833, U1112, U1247
tex commands:
\text_afterassignment:D ..... X49, X138, g2463
\text_aftergroup:D ... X57, X144, 965
\text_badness:D ..... S68, S74
\text_currentgrouplevel:D .....
    X48, X56, X137, X143
\text_deadcycles:D ..... X91
\text_endlinechar:D .....
    i447, g1737, g1835, g1836, g1842, 1151
\text_escapechar:D .... i202, i277,
    W212, W226, g1028, g1038, g1234,
    g1243, g1736, g2338, g2364, h1123, 252
\text_everypar:D ..... n20,
    n24, n29, n46, n50, n54, n69, n69,
    n109, n111, n121, n122, n173, n174, 351
\text_gdef:D ..... i192, i267, 312
\text_hskip:D ..... n95
\text_indent:D ..... n116
\text_kern:D ..... S65, S72
\text_lastnodetype:D ... n94, S58, 354
\text_lowercase:D ..... g2099, g2344
\text_marks:D ..... S146
\text_newlinechar:D ..... i448
\text_noindent:D .. n27, n52, n126, 350
\text_par:D ..... n18, n44,
    n97, n104, n128, n169, n170, n171, 354
\text_parskip:D ..... n25, n26, n51
\text_setbox:D ..... X50, X139
\text_shipout:D ... X126, X380, X510
\text_splitbotmarks:D ..... S82, 858
\text_splitfirstmarks:D ..... S94, 858
\text_splitmaxdepth:D ..... S50
\text_unskip:D ..... n90, S56, 354
\text_vbadness:D ..... S51
\text_vfuzz:D ..... S52, 857
\text_vss:D ..... X334
\text_xdef:D ..... i198, i273, 312
\text ..... 1082
text commands:
\l_text_case_exclude_arg_t1 .. aa626
\text_case_switch:nnnn ..... aa629
\text_declare_case_equivalent:Nn
    ..... aa631
\text_declare_lowercase_mapping:nn
    ..... aa635
\text_declare_lowercase_mapping:nnn
    ..... aa636
\text_declare_titlecase_mapping:nn
    ..... aa642
\text_declare_titlecase_mapping:nnn
    ..... aa643
\text_declare_uppercase_mapping:nn
    ..... aa649
\text_declare_uppercase_mapping:nnn
    ..... aa650
\text_lowercase:n ..... 1147
\textacutedbl ..... s915, s1153, E234, E235, E721, E971
\textascendercompwordmark ..... s854, E154, E685, E954
\textasciiaacute ..... s965, s1114, E236, E237, E722, E995
\textasciibreve ..... s913, s1152, E238, E239, E723, E968
\textasciicaron ..... s914, s1151, E240, E241, E724, E969
\textasciicircum .. s286, s558, s1088, 1116
\textasciidieresis ..... s953, s1101, E242, E243, E725, E985
\textasciigrave ..... s904, s1082, E244, E245, E726, E966
\textasciimacron ..... s960, s1109, E246, E247, E727, E990
\textasciitilde .. s287, s559, s1093, 1116
\textasteriskcentered ..... s267, s717, s864, s865, s1215, t164,
    t170, E120, E577, E642, E961, 1142
\textbackslash ..... s268, s560, s718, s1087, 1104
\textbaht ..... s939,
    s1156, E252, E253, E731, E1101, E1102
\textbar ..... s269, s561, s719, s1091, 1103
\textbardbl ..... s270,
    s720, s919, s1170, t169, E127, E365,
    E366, E578, E650, E797, E974, 1125
\textbf ..... D19, 574
\textbigcircle ..... s729, s892, s1232,
    E254, E255, E732, E1053, E1054, 1126
\textblank ..... s861,
    s1229, E345, E346, E785, E1023, E1024
\textborn ..... s905,
    E256, E257, E387, E733, E1059, E1060

```

File Key: a=ltDIRchk.dtx, b=ltPLAIN.dtx, c=ltVERS.dtx, d=ltLUATEX.dtx, e=ltEXPL.dtx,
f=ltDEFNS.dtx, g=ltCMD.dtx, h=ltHOOKS.dtx, i=ltCMDBOOKS.dtx, j=ltALLOC.dtx, k=ltCNTRL.dtx,
l=ltERROR.dtx, m=ltPAR.dtx, n=ltPARA.dtx, o=ltMETA.dtx, p=ltSPACE.dtx, q=ltLOGOS.dtx,
r=ltFILES.dtx, s=ltOUTENC.dtx, t=ltCOUNTS.dtx, u=ltLENGTH.dtx, v=ltFSSBAS.dtx,
w=ltFSSAXES.dtx, x=ltFSSTRC.dtx, y=ltFSSCMP.dtx, z=ltFSSDCL.dtx, A=ltFSSINI.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltFNTCMD.dtx, E=ltTEXTCOMP.dtx, F=ltPAGENO.dtx,
G=ltXRREF.dtx, H=ltMISCEN.dtx, I=ltMATH.dtx, J=ltLISTS.dtx, K=ltBOXES.dtx, L=ltTAB.dtx,
M=ltPICTUR.dtx, N=ltTHM.dtx, O=ltSECT.dtx, P=ltFLOAT.dtx, Q=ltIDXGLO.dtx, R=ltBIBL.dtx,
S=ltMARKS.dtx, T=ltPAGE.dtx, U=ltCLASS.dtx, V=ltKEYS.dtx, W=ltFILEHOOK.dtx,
X=ltSHIPOUT.dtx, Y=ltOUTPUT.dtx, Z=ltHYPHEN.dtx, aa=ltFINAL.dtx

```

\textbraceleft ..... s271, s308, s562, s721, s1090, 1103
\textbraceright ..... s272, s309, s563, s722, s1092, 1104
\textbrokenbar ..... s951, s1099, E128, E651, E983
\textbullet ..... s273, s723, s921, s1179, E121, E579, E643, E976
\textcapitalcompwordmark ..... s853, E153, E684, E953
\textcelsius ..... s922, s1198, E129, E353, E354, E652, E791, E977
\textcent .. s947, s1095, E130, E653, E980
\textcentoldstyle . s924, E258, E259, E392, E395, E734, E1075, E1076, 642
\textcircled ..... s279, s283, s300, s301, s730, s893, E155, E156, E655, E671, E687, E858, E1123, E1125, 1108
\textcircledP ..... s958, s1200, E260, E261, E735, E1117, E1118
\textcolonmonetary ..... s926, s1191, E313, E314, E765, E1077, E1078
\textcommaabove ..... s353, s355, s369, s370, s458, s459, s700, s701, 1129
\textcommabelow ..... s324, s326, s332, s333, s703, s704, s705, s706, s707, s708, s709, s710, s711, s712, s1252, s1455, s1456, s1457, s1458, 1129
\textcompsubstdefault ..... E34, E39, E91, E618, E916, 1136
\textcompwordmark ..... s290, s291, s564, s1158, 417
\textcopyright ..... s956, E262, E263, E368, E736, E1115, E1116
\textcopyright ..... s283, s317, s954, s1102, E131, E654, E986, 1121
\textcurrency ..... s949, s1097, E335, E336, E778, E853, E857, E1011, E1012
\textdagger ..... s275, s312, s725, s917, s1177, t165, t171, E123, E581, E645, E972, 1104
\textdaggerdbl ..... s274, s313, s724, s918, s1178, t166, t172, E122, E580, E644, E973, 1104
\textdblyhyphen ..... s876, E266, E267, E369, E738, E1025, E1026
\textdblyhyphenchar ..... s912, E264, E265, E370, E737, E1071, E1072
\textdegree . s961, s1110, E132, E656, E991
\textdied ..... s907, E268, E269, E388, E739, E1063, E1064
\textdiscount ..... s941, s1190, E270, E271, E740, E1105, E1106
\textdiv .. s978, s1135, E133, E657, E1005
\textdivorced ..... s906, s1235, E272, E273, E741, E1061, E1062
\textdollar ..... s255, s307, s438, s565, s798, s862, s1084, E114, E115, E623, E625, E959, E1131, E1133, 1103
\textdollaroldstyle s923, E274, E275, E393, E394, E742, E1073, E1074, 642
\textdong ..... s935, s1195, E315, E316, E766, E1095, E1096
\textdownarrow ..... s903, s1210, E317, E318, E767, E1057, E1058
\texteightoldstyle ..... s886, E214, E215, E384, E710, E1043, E1044
\textellipsis ..... s296, s321, s1180
\textemdash ..... s256, s407, s411, s566, s570, s785, s1162, 1103
\textendash ..... s257, s408, s410, s567, s569, s786, s1161, 1103
\textestimated ..... s942, s1206, E329, E330, E774, E856, E1009, E1010
\texteuro ..... s976, s1196, E351, E352, E789, E854, E1006, E1007
\textexcldown ..... s258, s412, s414, s571, s787, s1094, 1126
\textfiguredash ..... s410, s569, s1160, s1166, 1142
\textfiveoldstyle ..... s883, E216, E217, E381, E711, E1037, E1038
\textfloatsep ..... Y732, Y745, Y2142, Y2192, Y2341
\textflorin ..... s925, s1149, E333, E334, E777, E978
\textfont ..... x337, I251
\textfouroldstyle ..... s882, E218, E219, E380, E712, E1035, E1036
\textfraction ..... Y1954, Y1957, Y1982, Y1985, Y2134, Y2335, 1122
\textfractionsolidus ..... s877, s1187, E337, E338, E780, E962, 1122
\textgravedbl ..... s916, s1154, E248, E249, E728, E970
\textgreater . s281, s572, s740, s1086, 1112
\textguarani ..... s929, E276, E277, E391, E743, E1083, E1084
\textheight ..... r22, r23, r92, r93, r149, r150, P257, P258, P261, P287, P301, X380, Y78, Y227, Y228, Y276, Y401, Y449, Y478, Y649, Y708, Y767, Y819, aa156, aa157, 1094
\texthorizontalbar ..... s411, s570, s1163, s1168, 1142
\texthyphen ... s260, s417, s574, s789, 1109
\texthyphenchar s259, s416, s573, s788, 1109
\textindent ..... 1110

```

File Key: a=ltchapchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntr.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfnctcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltthyphen.dtx, aa=ltfinal.dtx

\textinterrobang s933,
 s1186, E349, E350, E787, E1091, E1092
 \textinterrobangdown s934,
 s1236, E347, E348, E786, E1093, E1094
 \textit D21
 \textlangle s888,
 s1227, E309, E310, E762, E1047, E1048
 \textlbrace 1103
 \textlbrackdbl
 s900, E210, E211, E389, E707, E964
 \textleaf s908,
 E278, E279, E373, E744, E1065, E1066
 \textleftarrow s859,
 s1207, E319, E320, E768, E1019, E1020
 \textlegacyasteriskcentered E589, E800
 \textlegacybardbl E589, E801
 \textlegacybullet E589, E802
 \textlegacydagger E589, E804
 \textlegacyparagraph E589, E803
 \textlegacyperiodcentered . E589, E806
 \textlegacysection E589, E807
 \textless . . . s280, s575, s739, s1085, 1116
 \textlira s931,
 s1192, E321, E322, E769, E1087, E1088
 \textlnot . . . s957, s1107, E134, E658, E988
 \textlquill s945,
 s1188, E280, E281, E745, E1111, E1112
 \textmacron 1123
 \textmarried s909,
 s1234, E282, E283, E746, E1067, E1068
 \textmd D19
 \textmho s891,
 s1205, E284, E285, E747, E1051, E1052
 \textminus
 s889, s1211, E343, E344, E783, E963
 \textmu s966, s1115, E341, E342, E782, E996
 \textmusicalnote s910,
 s1233, E286, E287, E748, E1069, E1070
 \textnaira s928,
 s1193, E288, E289, E749, E1081, E1082
 \textnineoldstyle s887,
 E220, E221, E385, E713, E1045, E1046
 \textnonbreakinghyphen
 s409, s568, s1159, s1164, 1142
 \textnormal D15
 \textnumero s940,
 s1199, E331, E332, E775, E1103, E1104
 \textogonekcentered s487, s698, s699, 1126
 \textohm s899,
 s1204, E339, E340, E781, E855, E1008
 \textonehalf s974, s1124, E135, E659, E1002
 \textoneoldstyle s879,
 E222, E223, E377, E714, E1029, E1030
 \textonequarter
 s973, s1123, E136, E660, E1001
 \textonesuperior s970, s1118,
 E137, E355, E356, E661, E792, E999
 \textopenbullet s943,
 s1231, E290, E291, E750, E1107, E1108
 \textordfeminine
 s305, s955, s1103, E138, E662, E987
 \textordmasculine
 s306, s971, s1119, E139, E664, E1000
 \TextOrMath
 t161, t164, t165, t166, t167, t168,
 t169, t170, t171, t172, t177, t184, 1128
 \textparagraph
 s276, s310, s726, s967, s1116,
 t168, E124, E582, E646, E997, 1104
 \textperiodcentered s277, s277,
 s968, s1117, E125, E583, E647, E998
 \textpermill 1122
 \textpertenmill 1122
 \textpertenthousand s492,
 s937, s1182, E306, E307, E308,
 E759, E1097, E1098, E1135, 1122
 \textperthousand . . . s490, s920, s1181,
 E118, E119, E639, E975, E1134, 1122
 \textpeso s930,
 s1197, E292, E293, E751, E1085, E1086
 \textpilcrow s938,
 E294, E295, E386, E752, E1099, E1100
 \textpm . . . s962, s1111, E140, E666, E992
 \textquestiondown s261,
 s413, s415, s576, s790, s1126, 1103
 \textquotedbl s579, s1083, 1103
 \textquotedblleft
 s262, s418, s577, s791, s1174, 1103
 \textquotedblright
 s263, s419, s578, s792, s1175, 1103
 \textquotelleft
 s264, s420, s580, s793, s1171, 1103
 \textquoteright
 s265, s421, s581, s794, s1172, 1103
 \textquotesingle
 s863, s1081, E141, E667, E960
 \textquotestraightbase
 s855, E142, E371, E668, E955
 \textquotestraightdblbase
 s856, E143, E372, E669, E956
 \texttriangle s890,
 s1228, E311, E312, E763, E1049, E1050
 \textrbrace 1103
 \textrbrackdbl
 s901, E212, E213, E390, E708, E965
 \textrecipie s932,
 s1201, E296, E297, E753, E1089, E1090

File Key: a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltruatex.dtx, e=ltxpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltthyphen.dtx, aa=ltfinal.dtx

\textreferencemark s969,
 s1185, E298, E299, E754, E1119, E1120
 \textregistered s300, s301,
 s959, s1108, E144, E670, E989, 1104
 \textrightarrow s860,
 s1209, E323, E324, E770, E1021, E1022
 \textrm D15
 \textrquill s946,
 s1189, E300, E301, E755, E1113, E1114
 \textsc D21, 1121
 \textsection s278,
 s311, s582, s728, s952, s1100, t167,
 E126, E584, E585, E648, E984, 1104
 \textservicemark s944,
 s1202, E302, E303, E756, E1109, E1110
 \textsevenoldstyle s885,
 E224, E225, E383, E715, E1041, E1042
 \textsf D15
 \textsixoldstyle s884,
 E226, E227, E382, E716, E1039, E1040
 \textsl D21
 \textssc w536, D31, D39
 textssc D25
 \textsterling s266, s319, s445,
 s583, s805, s948, s1096, E116, E117,
 E624, E632, E981, E1130, E1132, 1103
 \textstyle q15, B469, I63
 \textsubscript P419
 \textsuperscript s303, s305,
 s306, E663, E665, E679, P402, 1111
 \textsurd s972,
 s1226, E304, E305, E757, E1121, E1122
 \textsw w531, D30, D38
 textsw D25
 \TextSymbolUnavailable s3, s758, 1116
 \textthreeoldstyle s881,
 E228, E229, E379, E717, E1033, E1034
 \textthreequarters
 s975, s1125, E146, E675, E1003
 \textthreequartersemdash s858, E145,
 E357, E358, E375, E674, E793, E958
 \textthreesuperior s964, s1113,
 E147, E359, E360, E676, E794, E994
 \texttildelow
 s911, s1155, E250, E251, E729, E967
 \textttimes s977, s1129, E148, E677, E1004
 \texttrademark s303,
 s936, s1203, E149, E678, E979, 1116
 \textttt D15
 \texttwelveudash s857, E150,
 E361, E362, E374, E680, E795, E957
 \texttwooldstyle s880,
 E230, E231, E378, E718, E1031, E1032
 \texttwosuperior s963, s1112,
 E151, E363, E364, E681, E796, E993
 \textulc w526, D28, D29, D35, D37
 textulc D25
 \textunderline 1103
 \textunderscore
 s288, s315, s584, s1089, 1103
 \textup D21, 502
 \textuparrow s902,
 s1208, E325, E326, E771, E1055, E1056
 \textvisibleSPACE s292, s585, s1230, 1103
 \textwidth r24,
 r94, r151, K347, P266, Y79, Y146,
 Y203, Y220, Y634, Y644, Y693,
 Y703, Y2261, Y2293, aa157, 1094
 \textwon s927,
 s1194, E327, E328, E772, E1079, E1080
 \textyen s950, s1098, E152, E682, E982
 \textzerooldstyle s878,
 E232, E233, E376, E719, E1027, E1028
 \TH s535, s1131, aa655, 1099
 \th s586, s1137, aa655, 1099
 \thanks 803
 \thanks O10, O26, 1105
 \the 1083
 thebibliography (env.) 842
 \theenum 722
 \theequation I352, I364, I457, I518
 \thefootnote P396, P521, P526, P546
 \thempfn
 K357, P453, P458, P537, P542, P545
 \thempfootnote K357, P398
 \thepage r209,
 F6, G18, G35, G52, G106, G135,
 O164, O171, O177, Q15, Q32,
 R43, T14, Y246, Y277, Y1831, 977
 \Theta B299
 \theta B275
 \thetotalpages X350, X442, 977
 \thicklines M124
 \thickmuskip B646, I228, I230, I243
 \thickspace I214
 \thinlines M124, M816, M833
 \thinmuskip B644, I220, I222, I238, I244
 \thinspace p462,
 p468, p469, I189, I214, I251, 1137
 \thispagestyle T6
 \tilde B519
 \time a182, a186, b383
 \times B396
 \title 803
 \title O6, O7, O21, O23, O31
 \ commands:
 \c_empty_tl h65, h2110

File Key: a=ltirchk.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltxexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

```

\c_novalue_tl ..... g461, g840, g896, g957, g1382,
g1482, g1598, g1703, g1848, g1855,
g1895, g1947, g2028, g2055, g2137, 139
\c_space_t1 ..... i365, n66,
n75, W554, W557, aa605, aa610,
aa615, aa620, g24, g90, g107, g118,
g119, g135, g143, g153, g156, g158,
g160, g162, g170, g176, g226, g227,
g1082, g1097, g1117, g1118, g1390
\tl_clear:N ..... i196, i271, g241,
g286, g315, g382, g383, g599, g708,
g719, g720, g721, g723, g881, g949,
g1130, g1325, g1326, g1398, g1419,
g1509, g1510, g1526, g1738, g2087
\tl_const:Nn . i11, i12, X311, X316,
g1348, g2468, h35, h36, h110, h113,
h889, h890, h892, h893, h894, h895,
h898, h899, h901, h919, h920, h1256
\tl_count:N ..... W547, g333, g2111
\tl_count:n ..... g538, g543, g795, g902, g956, g1170
\tl_gclear:N ..... 277
\tl_gclear_new:N ..... h270
\tl_gput_left:Nn ..... V191
\tl_gput_right:Nn ..... n62, n71,
z320, g1057, g1318, g1320, h453, 261
\tl_gremove_once:Nn ..... h37, h37
\tl_gset:Nn ..... i124,
n15, n41, z284, z317, S81, S92,
S126, S127, S128, W51, h182, h194,
h301, h319, h415, h427, h443, h481,
h729, h781, h812, h815, h835, h838
\tl_gset_eq>NN ..... S79, S80, S85, S87, S91, S104,
S106, S115, S117, S119, S334, S336,
S338, S340, S342, S344, h63, h450, h461
\tl_head:N ..... g2401
\tl_if_blank:nTF ..... r521,
aa585, g534, g1492, g1909, g1950,
g2161, g2862, g2863, g2864, h433
\tl_if_empty:N ..... 283
\tl_if_empty:NTF ..... S83, S148,
g252, g253, g401, g649, g1332,
h235, h255, h364, h366, h630, h999,
h1380, h1914, h1921, h2093, h2095
\tl_if_empty:nTF ..... h2413, i120, i305, i324,
T30, T40, W40, W46, W58, W107,
W114, W116, W118, W417, g1906,
g2339, g2365, g2400, g2593, g2695,
h338, h351, h354, h800, h828, h880
\tl_if_empty_p:N ..... h2363, h2364, X68, X114, X378
\tl_if_empty_p:n ..... h857
\tl_if_eq:NNTF ..... S171, S178, g275
\tl_if_eq:nnTF ..... g661, g915, g2344
\tl_if_exist:N ..... 278
\tl_if_exist:NTF ..... h2242, h2386, h2427, g1333,
h135, h233, h253, h1234, h2138, h2158
\tl_if_exist_p:N ..... h276
\tl_if_head_eq_charcode:nTF ..... r523, r528
\tl_if_head_is_group:ntF ..... i332, g2159, g2203, g2235
\tl_if_head_is_N_type:nTF ..... i329, g2171, g2183, g2200, g2232
\tl_if_in:NnTF ..... g1523
\tl_if_in:nnTF ..... g550, g1489, g1515, 167
\tl_if_novalue:nTF ..... g306, g325,
g372, g885, g1979, g2854, g2855,
g2856, g2857, g2858, g2859, h332
\tl_if_single:nTF ..... g2076, g2837
\tl_if_single_token:nTF ..... i315, g609, g2336, g2839, 189
\tl_item:Nn ..... g2296
\tl_log:n ..... h37, h39, h1752
\tl_map_function:nN ..... g326, g535, g536, g900, g950, g2878
\tl_map_inline:Nn ..... g659
\tl_map_inline:nn ..... g1361, g1595
\tl_new:N i6, i8, i9, i10, n14, S24, S25,
S26, S27, S28, S29, S30, S31, S32,
S33, S34, S35, S36, S37, S38, S39,
S40, S41, S44, S45, W8, W9, W10,
W11, W59, X53, X205, g11, g12,
g13, g14, g17, g21, g28, g29, g30,
g33, g38, g39, g41, g42, g50, g51,
g1504, g1505, g1721, g2073, g2303,
g2873, h25, h26, h27, h29, h32, h78,
h96, h124, h138, h141, h165, h166,
h299, h317, h1253, h1491, h1492, h1493
\tl_put_left:Nn ... g761, g770, g1417
\tl_put_right:Nn .. i168, i182, i243,
i257, aa626, g301, g327, g471, g504,
g527, g541, g558, g563, g691, g786,
g799, g826, g834, g848, g850, g857,
g874, g890, g896, g965, g990, g1145,
g1153, g1169, g1176, g1393, g1396,
g1513, g1514, g1521, g1531, g1599,
g1633, g1860, g1870, g2089, g2134
\tl_replace_all:Nnn ..... g2102
\tl_rescan:nn . i20, i20, i462, i543, i571
\tl_set:Nn ..... i166, i179, i180, i183, i185, i206,
i241, i254, i255, i258, i260, i281,
i373, i378, i459, i509, i540, i568,

```

File Key: a=ltirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntr1.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfnctcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

W81, W82, W83, W84, X47, X136, X210, X236, X261, X288, g22, g88, g89, g242, g243, g244, g245, g273, g324, g805, g1002, g1020, g1209, g1389, g1424, g1486, g1592, g1619, g1626, g1639, g1645, g1651, g1657, g1663, g1669, g1675, g1681, g1712, g1734, g1757, g2069, g2070, g2096, g2120, g2151, g2163, g2190, g2256, g2258, g2270, g2296, g2438, g2439, h37, h40, h554, h1108, h1115, h1119, h1500, h1520, h1521, h1526, h1527, h1542, h1549, h1550, h1581, h1601, h1602, h1607, h1608, h1623, h1630, h1631, h1677, h1684
 \tl_set_eq:NN ... i197, i272, g263, g269, g277, g320, g1418, g1525, g1598, h1530, h1553, h1611, h1634
 \tl_show:N g2319, g2326, 177
 \tl_show:n ... g1250, g1263, g1272, g1294, h37, h38, h1757, h1858, h1953
 \tl_tail:N g1791
 \tl_to_str:N g1846, g1853
 \tl_to_str:n
 . i113, i118, i457, i476, i510, i521, i557, S144, S157, W581, g70, g81, g186, g190, g396, g404, g417, g428, g432, g468, g499, g580, g604, g612, g627, g640, g652, g671, g705, g1143, g1217, g1223, g1393, g1397, g1847, g1854, g2023, g2050, g2130, g2131, g2312, g2325, g2353, g2357, h382, h531, h602, h1904, h1916, h1981, 165
 \tl_trim_spaces:n
 . i307, W41, W42, g182, g353, g471, g505, g1407, g1409, g2022, g2049, g2151, g2476, h411, 312
 \tl_trim_spaces_apply:nN
 . g609, g2154, g2178, g2330, h334
 \tl_use:N ... z288, z292, z327, z331, g1287, g1338, h1016, h1079, h1743
 tl internal commands:
 \g__mark_new_top_tl
 . S43, S79, S80, S86, S88
 \g__mark_tmp_tl S43, S81, S83, S91, S141, S144, S148, S150, 858
 \tmspace I214, 699
 \to B440, B442
 \today ... a187, a191, a199, a202, O33, 1075
 token commands:
 \c_space_token g2428
 \token_if_active:NTF g1862
 \token_if_cs:NTF g2078, 179
 \token_if_eq_catcode:NNTF i317, g1543
 \token_if_eq_charcode:NNTF
 . g348, g1779, g1811, g1822
 \token_if_eq_meaning:NNTF ... i187, i262, i463, W455, W467, g618, g621, g629, g1210, g2442, g2841, g2843
 \token_if_macro:NTF
 . i84, h1163, h1189, 306
 \token_if_math_toggle_p:N
 . g2224, g2246
 \token_if_protected_macro:NTF
 . g1085, g1264
 \token_to_meaning:N
 . i80, i460, i472, i541, i569, n65
 \token_to_str:N i79, i80, i102, i527, i534, i562, i565, W241, g70, g78, g81, g353, g739, g997, g1138, g1286, g1296, g1556, g1702, g1863, g2022, g2049, g2340, g2353, g2357, g2366, g2477, g2747, g2748, g2761, g2762, g2803, g2804, g2817, g2818, h379, 178
 \toks n64, n73, d36, z668, z669, z679, z688, aa703, b31, b63, b95, 352
 \toksdef d230, b46, b63, b95
 \tokszero d230
 \tolerance v672, v717, v732, T87, T95, b317
 \top B320
 \topfigrule Y731, Y2363
 \topfraction P273, Y2329
 \topmargin Y71, Y628, Y687
 \TopMark S274, S379, 852
 \topmark Y2247, Y2256, 847
 \topsep I524, J2, J59, 711
 \topskip r59, r128, r183, J1, Y130, b418, 1113
 \totalheight K33, K34, K35
 totalpages 961
 trace\check@cr commands:
 trace\string_stack\string_levels aa96
 tracefloats Y1933
 tracefloatsoff Y1933
 tracefloatvals Y1933, 1043
 traceoff 338
 traceon 338
 tracingall b557, 338
 tracingassigns ... b575, b609, b626, b667
 tracingcommands b573, b591, b607, b616, b629, b653, b670, b347
 tracingfonts x17, x54, x58, x86, x118, x153, x194, x224, x238, x254, x260, x273, x280, x287, x292, x301, x314, x322, x325, 1081
 tracinggroups ... b565, b600, b638, b678
 tracingifs b566, b601, b637, b677

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\tracinglostchars . b558, b563, b586, b598, b617, b641, b661, b681, b346, 25
 \tracingmacros b572, b590, b606, b618, b640, b660, b680, b341
 \tracingnesting .. b568, b603, b635, b675
 \tracingnone b622
 \tracingoff x118, x322
 \tracingon x119, x323, 1081
 \tracingonline b552, b628, b652, b669, b699, v681, Y1919, b340
 \tracingoutput b553, b632, b656, b673, b696, b345
 \tracingpages b562, b585, b597, b617, b642, b662, b682, b344
 \tracingparagraphs b564, b587, b599, b618, b639, b659, b679, b343
 \tracingrestores b574, b592, b608, b618, b627, b658, b668, b348
 \tracingscantokens b567, b582, b602, b636, b650, b676
 \tracingstacklevels b558, b570, b634, b349, 25
 \tracingstats b561, b584, b596, b616, b643, b663, b683, aa2, b342
 \triangle B322
 \triangleleft B358, B482
 \triangleright B359, B482
 \TrimSpaces g2874
 trivlist (env.) J89
 \trivlist ... H358, H408, H410, H424, H446, I508, J89, L78, N35, N37, 718
 \tt 1085
 \ttdefault A12, A216, A400, A417, A441, A460, A488, B50
 \ttfamily ... A10, A11, A410, A458, A459, A486, A487, D17, H468, 217
 ttfamily A420
 \ttsubstdefault ... B20, B32, E32, E89
 \twocolumn Y201, 1119
 \twocolumn[...] 1114
 \twocolumn[] 384
 \typein 82, 82
 \typein f31
 \typeout 82
 \typeout l74, c21, c38, a119, r201, r673, r674, r680, r714, r752, r766, r769, v391, w478, a175, A138, A563, A744, A754, A764, B9, B125, a200, a202, a214, a229, a236, a247, f3, Q8, Q25, S239, S240, S244, S248, S252, S256, S260, f36, f43, a260, U323, U339, U351, a273, U1483, U1682, U1685, a286, X104, X115, X148, Y1922, Y1934, a324, aa276, f614, f615, aa657, aa664, aa676, aa677, aa685, f649, f650, f655, f709, f710, f725, f727, a76, 39

U

\u s237, s386, s472, s589, s596, s616, s623, s754, s1242, s1317, s1318, s1333, s1334, s1343, s1344, s1357, s1358, s1359, s1383, s1384, s1409, s1410, E163, E194
 \uccode aa240, aa248, aa255, aa257, aa260, aa262, aa540, aa548, aa555, aa557, aa560, aa562
 \Ucharcat A624
 \uchyph b370
 \ulcdefault w526, w605
 \ulcshape w526, w604, w698, w699, A549, D29, 501
 \Umathcode p478, d30, B15, B50, B65, E159, E367, H474, aa160, aa351, aa534, b127
 \unboldmath A615
 \UndeclareTextCommand s191, E115, E117, E119, E308, E585, E1130, E1131, E1132, E1133, E1134, E1135, 1122
 \undefined t71, t72, t128, t129, t130, t131, t132, t133, B116, B117, a12, G40, G41, G57, G58, G59, G60, a14, a20, a168, aa180, aa181, aa201, a60, 1115
 \undefinedpagestyle T4, T8
 \underbar f893, f914, b522
 \underbrace B540
 \underline 726
 \underline K454, K455, b522
 \underscore 1121
 \unexpanded .. D47, H198, H244, H261, U742, U1511, U1513, f632, f643, 308
 \unhbox 1082
 \unhcop y L345, M742, M798, b524
 \unicodedataline d143, d146, d160, d161, d162
 \UnicodeEncodingName s981, s987, s1038, s1044, s1048, s1059, s1063, s1079, s1080, E376, E377, E378, E379, E380, E381, E382, E383, E384, E385, E386, E387, E388, E389, E390, E391, E392, E393, 437
 \UnicodeFontFile s1036
 \UnicodeFontName s1037
 \UnicodeFontTeXLigatures s993, s1033
 \unicoderead d143, d157, d158, d159, d160, d165

File Key: a=ltdirname.dtx, b=lpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx, f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx, l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx, r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx, w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx, B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx, G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx, M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx, S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx, X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphe n.dtx, aa=ltfinal.dtx

uninstall [d995](#)
\unitlength [K53](#), [K64](#), [K73](#), [K83](#),
[M5](#), [M29](#), [M30](#), [M32](#), [M34](#), [M42](#),
[M43](#), [M44](#), [M45](#), [M60](#), [M63](#), [M73](#),
[M74](#), [M84](#), [M85](#), [M93](#), [M94](#), [M107](#),
[M108](#), [M119](#), [M164](#), [M176](#), [M241](#),
[M256](#), [M316](#), [M318](#), [M332](#), [M340](#),
[M342](#), [M357](#), [M375](#), [M377](#), [M392](#),
[M397](#), [M399](#), [M414](#), [M417](#), [M422](#),
[M480](#), [M481](#), [M508](#), [M509](#), [M537](#),
[M538](#), [M612](#), [M628](#), [M648](#), [M654](#),
[M690](#), [M691](#), [M693](#), [M694](#), [M697](#),
[M698](#), [M700](#), [M701](#), [M712](#), [M713](#),
[M715](#), [M716](#), [M718](#), [M719](#), [M721](#),
[M722](#), [M750](#), [M751](#), [M753](#), [M754](#),
[M757](#), [M758](#), [M760](#), [M761](#), [M772](#),
[M774](#), [M776](#), [M778](#), [X328](#), [aa149](#), [957](#)
\unkern [v700](#)
\unless [d151](#), [d159](#), [d161](#)
\unlhd [A732](#)
\unpenalty
 . [v703](#), [v707](#), [D116](#), [H440](#), [H462](#), [348](#)
\unrhd [A734](#)
\unsetattribute [44](#)
\unsetattribute [d82](#), [d240](#)
\unskip [1134](#)
\unvcopy [I193](#), [854](#)
\Uparrow [B584](#)
\uparrowarrow [B578](#)
\upbracefill [B543](#), [B560](#)
\updefault [w688](#),
 A21, [B94](#), [B101](#), [B103](#), [B111](#), [B113](#), [604](#)
\Updownarrow [B588](#)
\updownarrow [B582](#)
\uplus [B378](#)
\uppercase [1095](#)
\upshape [s442](#), [s732](#), [s802](#),
[s895](#), [w686](#), [w687](#), [A19](#), [A20](#), [A549](#),
[A560](#), [A593](#), [A651](#), [D24](#), [E629](#), [501](#)
\Upsilon [B304](#)
\upsilon [B286](#)
 use commands:
 \use:N . [h2204](#), [h2270](#), [h2285](#), [z295](#),
[z334](#), [e178](#), [S82](#), [S95](#), [S146](#), [W443](#),
[W447](#), [W449](#), [W466](#), [aa596](#), [g735](#),
[g1335](#), [h1569](#), [h1693](#), [h1694](#), [h1734](#)
 \use:n
 . [h2254](#), [h2259](#), [h2267](#), [i105](#), [i172](#),
[i175](#), [i203](#), [i212](#), [i247](#), [i250](#), [i278](#),
[i284](#), [i350](#), [i383](#), [i406](#), [i425](#), [i454](#),
[i484](#), [i511](#), [i518](#), [i554](#), [r523](#), [V136](#),
[g140](#), [g619](#), [g622](#), [g793](#), [g1026](#),
[g1140](#), [g1232](#), [g1303](#), [g2211](#), [g2253](#),
 \usebox [K156](#)
 \usecounter [J225](#), [J238](#)
 \usefont [s1570](#), [v80](#), [v282](#),
[v677](#), [A709](#), [E7](#), [E613](#), [H478](#), [1114](#)
 \UseHook [h2699](#), [h2815](#), [i290](#),
[i562](#), [i565](#), [r316](#), [r322](#), [r327](#), [r338](#),
[r377](#), [r381](#), [r384](#), [x145](#), [A263](#), [A274](#),

File Key: a=ltdirchk.dtx, b=ltpplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspace.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=lxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=lt pictur.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=lt hyphen.dtx, aa=ltfinal.dtx

A286, A296, A317, A324, A337,
 A344, A403, A408, A413, A418,
 A663, A680, H180, H192, H195,
 H219, H222, H226, H229, U918,
 U922, U946, U950, W170, W171,
 W174, W175, X118, X170, X382, 210
 \backslash UseHookWithArguments
 .. h2699, i219, i527, i534, G104, 1151
 \backslash UseLegacyTextSymbols E576, E799
 \backslash UserName 79
 \backslash UserName e174, e192
 \backslash UseOneTimeHook h2699, h2816, r15, r57,
 r70, r317, r321, r326, r339, r378,
 r380, r383, H14, H18, H28, H29,
 H31, U919, U923, U945, U949, 194
 \backslash UseOneTimeHookWithArguments h2699, 194
 \backslash usepackage U613, U675, U1498, 199
 \backslash UseRawInputEncoding aa381, aa437, aa484
 \backslash UseTextAccent s149, s150, s188, E110,
 E111, E113, E156, E158, E939,
 E1124, E1125, E1127, E1128, 1103
 \backslash UseTextSymbol s150,
 s186, E109, E352, E938, E1007, 1103
 \backslash usetikzlibrary 199
 \backslash ushape 1122

V

\backslash v s238, s387, s471, s592,
 s593, s594, s598, s600, s603, s605,
 s607, s613, s619, s620, s621, s625,
 s627, s630, s632, s634, s640, s755,
 s1247, s1327, s1328, s1329, s1330,
 s1339, s1340, s1373, s1374, s1379,
 s1380, s1391, s1392, s1399, s1400,
 s1403, s1404, s1426, s1427, s1428,
 s1429, s1430, s1431, s1432, s1433,
 s1434, s1435, s1436, s1445, s1446,
 s1447, s1448, s1451, s1452, E165, E195
 \backslash vadjust p38, p60, p72, p101, p108, p342,
 p358, p376, p392, P201, P223, 342
 \backslash valign E101, E930
 \backslash value 449
 \backslash value t14, R9, 1082
 \backslash varbigtriangledown B362, B365
 \backslash varbigtriangleup B363, B364
 \backslash varepsilon q15, B291
 \backslash varphi B296
 \backslash varpi B293
 \backslash varrho B294
 \backslash varsigma B295
 \backslash vartheta B292
 \backslash vbadness X212,
 X214, X263, X265, Y2244, b319, 973
 \backslash vbox 1109

vbox commands:
 \backslash vbox_set_split_to_ht:NNn S76
 \backslash vbox_set_to_ht:Nnn
 S53, S62, X215, X266
 \backslash vbox_to_zero:n X326
 \backslash vbox_unpack:N
 S64, S69, S287, S318, X226, X276
 \backslash vdash B404
 \backslash vdots B510
 \backslash vec B524
 \backslash vector l269, M233, M817, M834
 \backslash vee B367, B369
 \backslash verb H472, H497, H510, H516, H525,
 H531, H541, H546, H559, H561, 1083
 \backslash verb* 1134
 \backslash verbatim (env.) H466
 \backslash verbatim H466
 \backslash verbatim* (env.) H490
 \backslash verbvisiblespace H472,
 H474, H481, H485, H497, H502, 1134
 \backslash Vert B571, B573
 \backslash vert B576
 \backslash vfil E102, E105, E931, E934,
 M566, M578, X393, X405, Y177,
 Y196, Y414, Y461, Y631, Y690, b511
 \backslash vfilneg b511
 \backslash vfuzz T90, T97,
 X211, X213, X262, X264, b391, 973
 \backslash vglue b501
 \backslash vline L366
 \backslash voffset b407
 \backslash vphantom s497, s513, I75
 \backslash vrule p454, s293, s295, s502, s518,
 x190, B558, B559, B561, B562,
 K165, K167, K218, K225, K453,
 K497, L186, L219, L347, L366,
 M227, M299, M302, M324, M333,
 M350, M359, M383, M391, M406,
 M413, M565, M578, M726, M782,
 Y1855, Y2264, Y2297, b505, 1053
 \backslash vsizer 1117
 \backslash vskip 1082
 \backslash vspace p331, p401, p402, p403
 \backslash vsplit Y384, Y431, Y2246

W

\backslash wedge B366, B368
 \backslash whatsit d194, 44
 \backslash widehat B527
 \backslash widetilde B526
 \backslash widowpenalties b106
 \backslash widowpenalty v690, b326
 \backslash width K30, 1102

File Key: a=ltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltluatex.dtx, e=ltexpl.dtx,
 f=ltdefns.dtx, g=ltcmd.dtx, h=lthooks.dtx, i=ltcmddhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
 l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspage.dtx, q=ltlogos.dtx,
 r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
 w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
 B=fontdef.dtx, C=preload.dtx, D=ltfnctcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
 G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
 M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
 S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
 X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx

\wlog	a103, d6, d7, d8, d54, H41, U353, aa60, aa710, <u>b40</u> , b145, b239, b254, b284, b299, 39	aa193, aa194, aa195, aa196, aa201, aa206, aa207, aa208, aa209, aa210, aa211, aa212, aa213, aa214, aa215, aa216, aa217, aa218, aa219, aa220		
\wp	B312	\XeTeXmathcode aa161, aa535		
\wr	B382	\XeTeXrevision aa41		
\write	1117	\XeTeXuseglyphmetrics aa270, aa272		
X				
\x	c96, c99, v333, v334, aa330, aa332	\XeTeXversion A623, aa41		
\xdef	1081	\Xi B301		
\XeTeXcharclass	v665, aa39, aa47, aa54, aa67, aa73, aa82, aa89	\xi B281		
\XeTeXcharclassCL	aa173	\xpt 1079		
\XeTeXcharclassCM	aa177	\xspacekip b422		
\XeTeXcharclassEX	aa174	\xtxHanGlue aa180, aa204, aa212, aa213, aa214, aa215, aa216, aa217, aa218, aa219, aa220		
\XeTeXcharclassID	aa171	\xtxHanSpace aa181, aa205, aa206, aa207, aa208, aa209, aa210, aa211		
\XeTeXcharclassIS	aa175	Y		
\XeTeXcharclassNS	aa176	\y c97, c99		
\XeTeXcharclassOP	aa172	\year c14, a188, U1192, U1325, U1414, b386		
\XeTeXcharglyph	s1034	Z		
\XeTeXdashbreakstate	aa273	\Z aa252, aa531, aa552		
\XeTeXglyph	s1034	\z aa243, aa532, aa543		
\XeTeXintercharclasses	aa167, aa200	\zeta B273		
\XeTeXinterchartoks	aa168, aa182, aa183, aa184, aa185, aa186, aa187, aa188, aa189, aa190, aa191, aa192,			

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=ltcmhooks.dtx, j=ltalloc.dtx, k=ltcntrl.dtx,
l=lterror.dtx, m=ltpar.dtx, n=ltpara.dtx, o=ltmeta.dtx, p=ltspare.dtx, q=ltlogos.dtx,
r=ltfiles.dtx, s=ltoutenc.dtx, t=ltcounts.dtx, u=ltlength.dtx, v=ltfssbas.dtx,
w=ltfssaxes.dtx, x=ltfsstrc.dtx, y=ltfsscmp.dtx, z=ltfssdcl.dtx, A=ltfssini.dtx,
B=fontdef.dtx, C=preload.dtx, D=ltfntcmd.dtx, E=lttextcomp.dtx, F=ltpageno.dtx,
G=ltxref.dtx, H=ltmisen.dtx, I=ltmath.dtx, J=ltlists.dtx, K=ltboxes.dtx, L=lttab.dtx,
M=ltpicture.dtx, N=ltthm.dtx, O=ltsect.dtx, P=ltfloat.dtx, Q=ltidxglo.dtx, R=ltbibl.dtx,
S=ltmarks.dtx, T=ltpage.dtx, U=ltclass.dtx, V=ltkeys.dtx, W=ltfilehook.dtx,
X=ltshipout.dtx, Y=ltoutput.dtx, Z=ltyphephen.dtx, aa=ltfinal.dtx